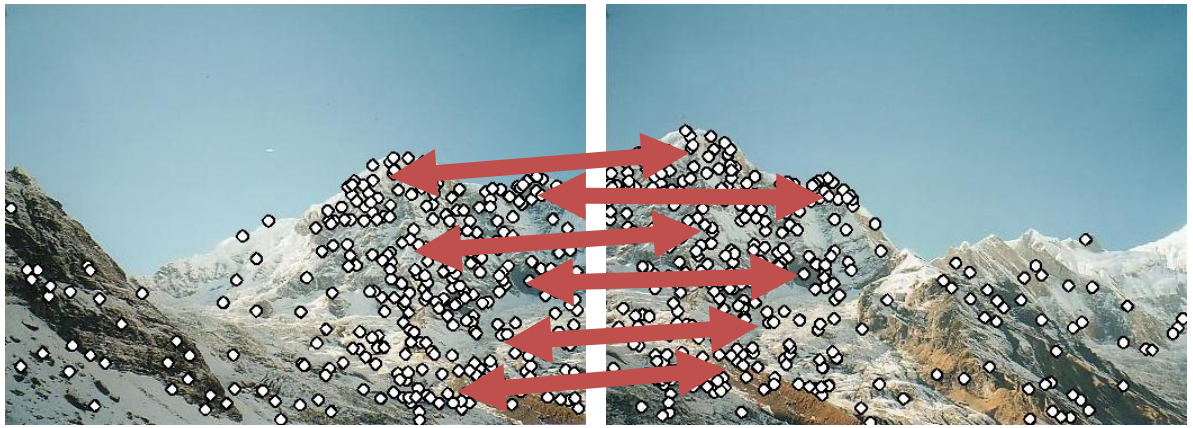
Transformations and Fitting

EECS 442 – Jeong Joon Park
Winter 2024, University of Michigan

So Far



1. How do we find distinctive / easy to locate features? *(Harris/Laplacian of Gaussian)*
2. How do we describe the regions around them? *(histogram of gradients)*
3. How do we match features? (L2 distance)
4. How do we handle outliers? (RANSAC)

Today

As promised: warping one image to another

Why Mosaic?

- Compact Camera FOV = 50 x 35°

Slide credit: Brown & Lowe

Why Mosaic?

- Compact Camera FOV = 50 x 35°
- Human FOV = 200 x 135°

Slide credit: Brown & Lowe

Why Mosaic?

- Compact Camera FOV = 50 x 35°
- Human FOV = 200 x 135°
- Panoramic Mosaic = 360 x 180°

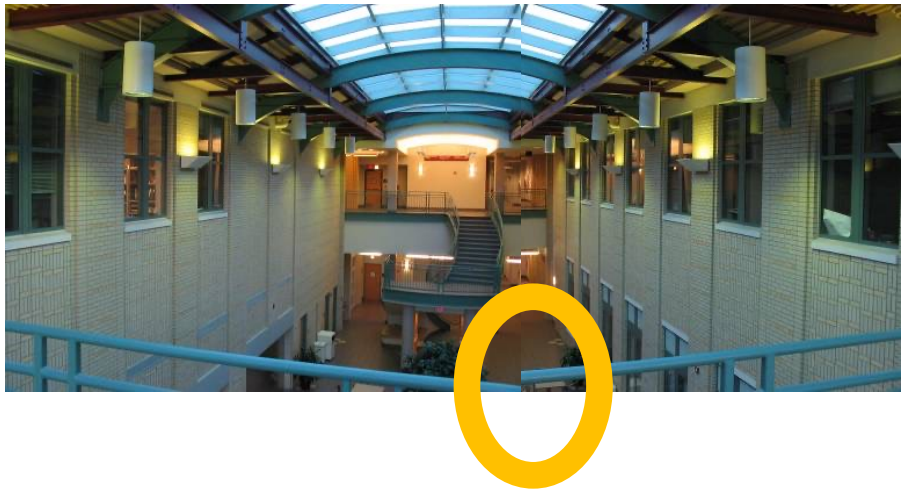
Slide credit: Brown & Lowe

Why Bother With This Math?

Slide credit: A. Efros

Homework 1 Style

Translation only via alignment



Slide credit: A. Efros

Result

Slide credit: A. Efros

Image Transformations

Image filtering: change range of image

$$g(x) = T(f(x))$$

intensity

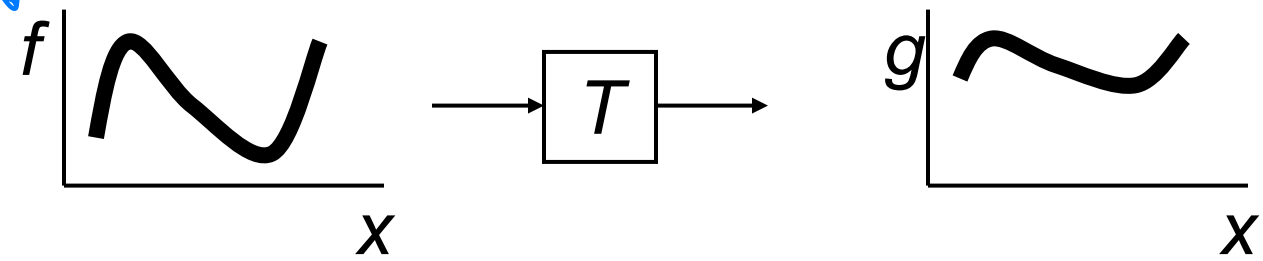
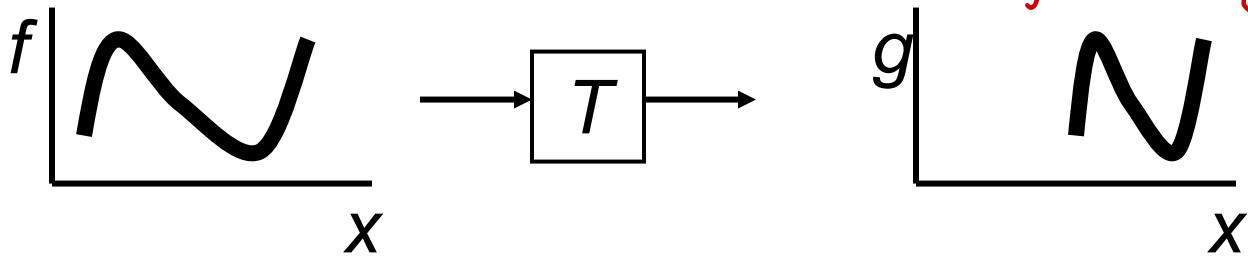


Image warping: change ***domain*** of image

$$g(x) = f(T(x))$$

change both the intensity and the domain of the image



Slide credit: A. Efros

Image Transformations

Image filtering: change range of image

$$g(x, y) = T(f(x, y))$$

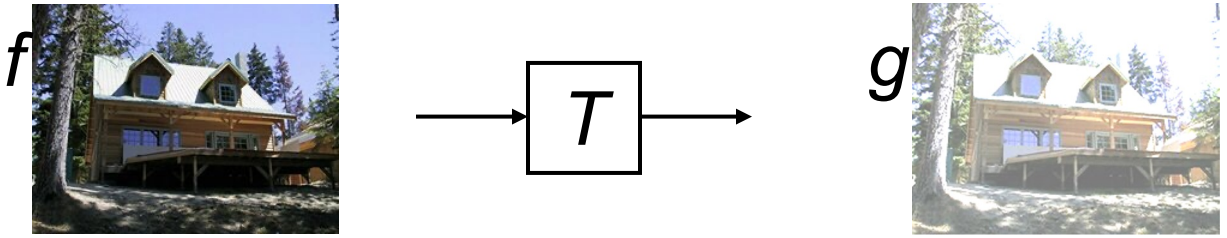
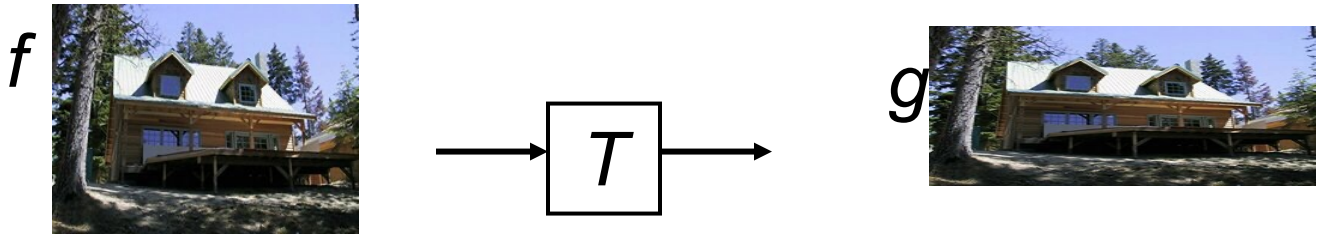


Image warping: change ***domain*** of image

$$g(x, y) = f(T(x, y))$$



Slide credit: A. Efros

Parametric (Global) warping

Examples of parametric warps

(parametric global warping)

參數

translation

moving this image as a whole, so you don't discriminate one pixel from another pixel. But rather you apply this transformation to all pixels in this images.

rotation

- extending
- compressing

aspect

affine

hence global warping

perspective

cylindrical

combination of these and shear

transforming an image in a way that kind of simulates the change of camera viewpoint

Slide credit: A. Efros

Parametric (Global) Warping

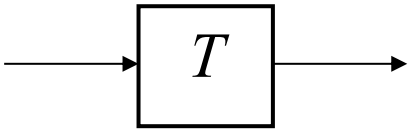
T is a coordinate changing machine

$$\boldsymbol{p}' = T(\boldsymbol{p})$$

a new coordinate.

Note: T is the same for all points, has relatively few parameters, and does **not** depend on image content

p = (x,y)



p' = (x',y')

Slide credit: A. Efros

Parametric (Global) Warping

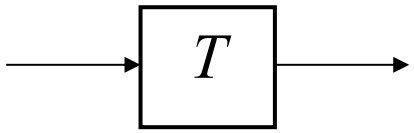
Homogenous coordinate.

Today we'll deal with linear warps

$$\boldsymbol{p}' \equiv \boldsymbol{T}\boldsymbol{p}$$

The transformations work in a homogenous coordinate space.

T: matrix; p, p': 2D points. Start with normal points and =, then do homogeneous cords and ≡



p = (x,y) 2D points

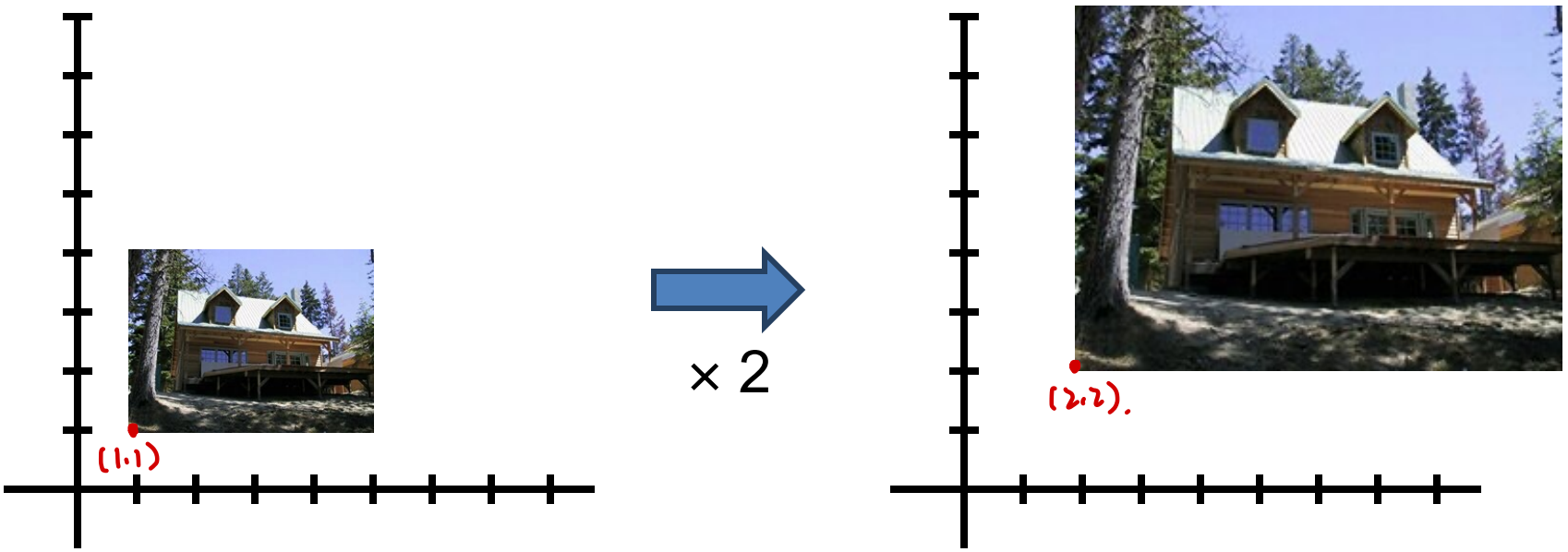
p' = (x',y')

Slide credit: A. Efros And oftentimes is 3D since Homogenous coordinate system.

Scaling

Scaling multiplies each component (x,y) by a scalar.
Uniform scaling is the same for all components.

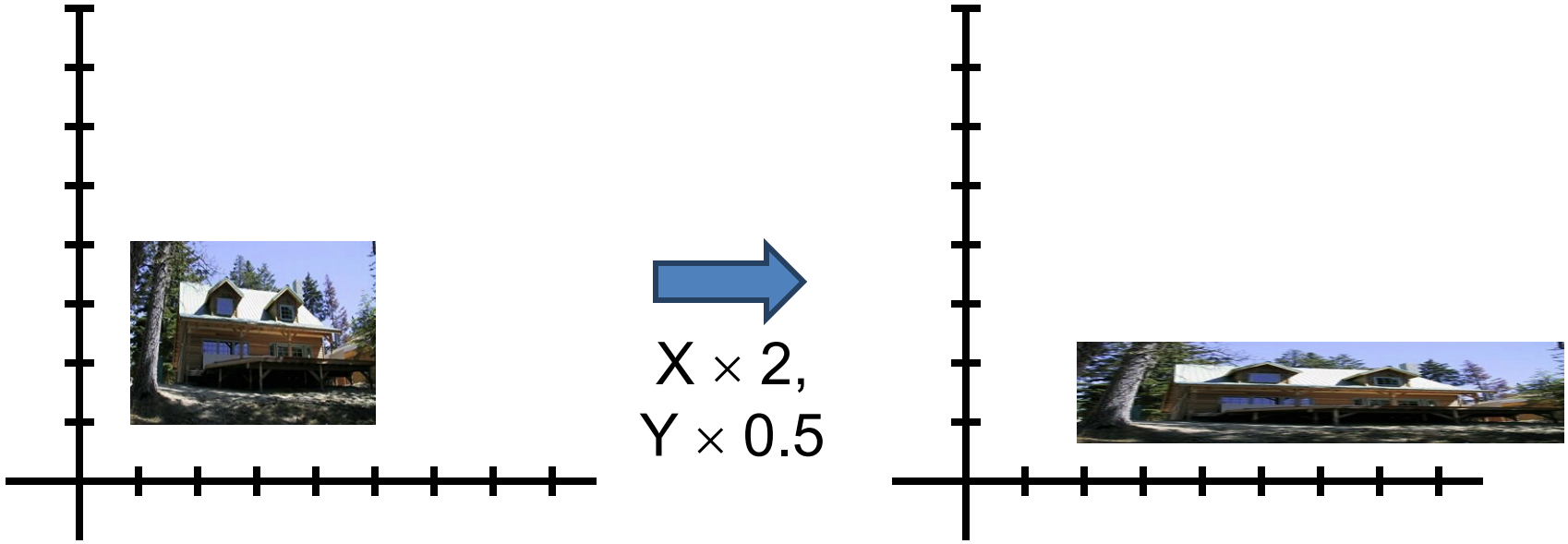
Note the corner goes from (1,1) to (2,2)



Slide credit: A. Efros

Scaling

Non-uniform scaling multiplies each component by a different scalar.



Slide credit: A. Efros

Scaling

What does T look like?

$$\begin{aligned} x' &= ax \\ y' &= by \end{aligned}$$

what we want

Let's convert to a matrix:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}} \begin{bmatrix} x \\ y \end{bmatrix}$$

$ax + 0y + 0x + by$

$= ax + by.$

scaling matrix S

What's the inverse of S?

?

Slide credit: A. Efros

2D Rotation

Rotation Matrix

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

But wait! Aren't sin/cos non-linear?

x' *is* a linear combination/function of *x, y*
x' *is not* a linear function of θ

What's the inverse of R_θ? $\boldsymbol{I = R_\theta^T R_\theta}$

Slide credit: A. Efros

Things You Can Do With 2x2

Identity / No Transformation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

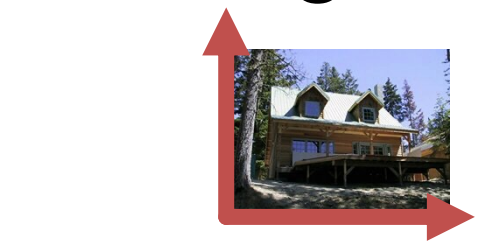
Shear

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & sh_x \\ sh_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Slide credit: A. Efros

Things You Can Do With 2x2

Before



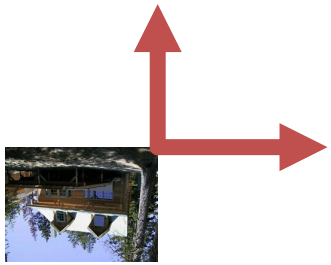
After

2D Mirror About Y-Axis

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Before

After



2D Mirror About X,Y

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Slide credit: A. Efros

What's Preserved? (Projection)

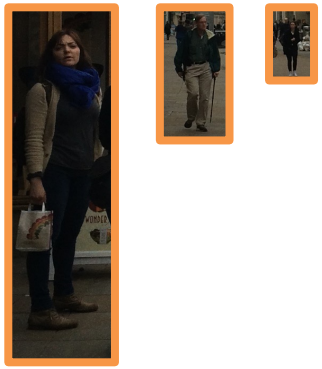
1. 3D lines project to 2D lines so lines are preserved

Projections of parallel 3D lines are not necessarily parallel, so not parallelism

2. parallelism is not preserved.

Distant objects are smaller

3. so size is not preserved



What's Preserved With a 2x2

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = T \begin{bmatrix} x \\ y \end{bmatrix}$$

After multiplication by T (irrespective of T)

- Origin is origin: **0 = T0**
- Lines are lines
- Parallel lines are parallel

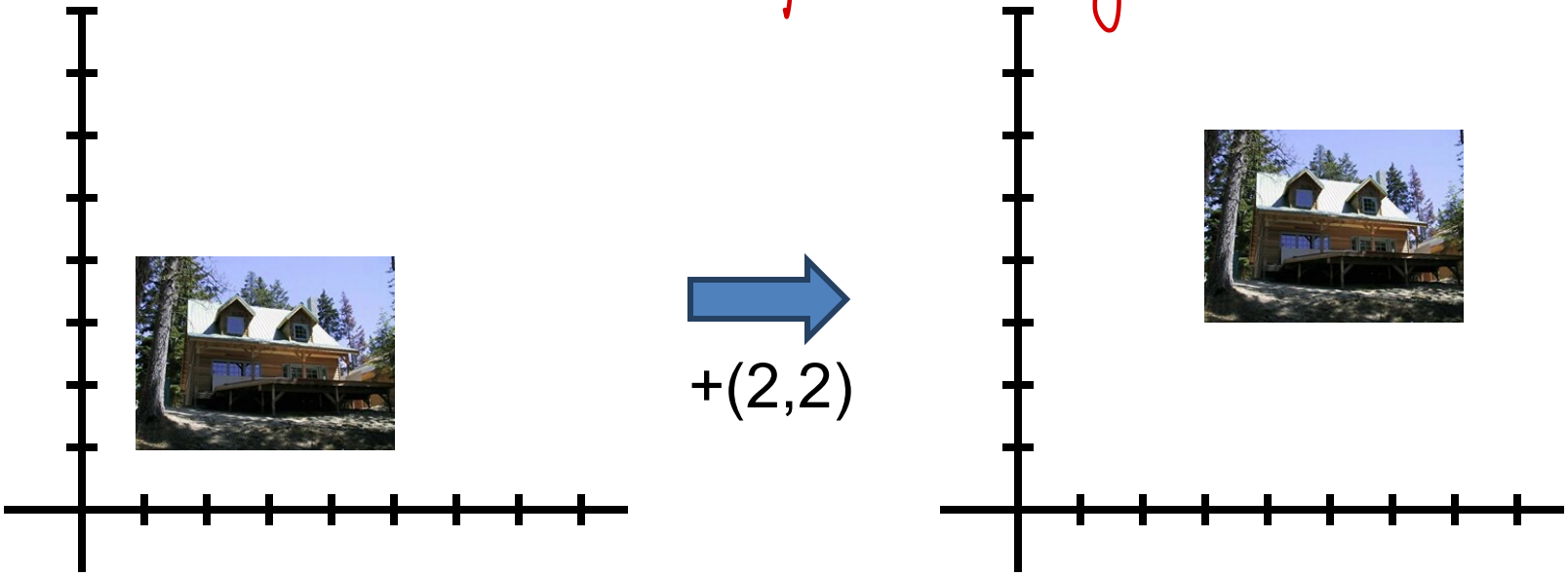
Things You Can't Do With 2x2

What about translation?

$x' = x + t_x$, $y' = y + t_y$

How do we make it linear?

we can't do this operation By a 2x2 matrix.



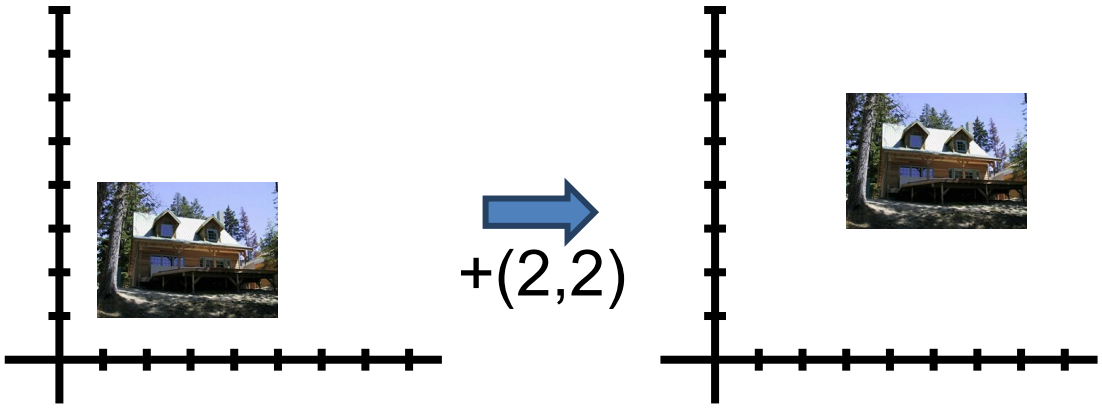
Homogeneous Coordinates Again

What about translation?

$x' = x + t_x$, $y' = y + t_y$

$$\begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix} \equiv \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \equiv \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

3x3



Slide credit: A. Efros

Representing 2D Transformations

How do we represent a 2D transformation?

Let's pick scaling

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \equiv \begin{bmatrix} s_x & 0 & a \\ 0 & s_y & b \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

What's a b d e f

0 0 0 0 1

Affine Transformations

Affine: *linear transformation* plus *translation*

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \equiv \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

transformation matrix

translation operation

2x2 linear transformation matrix

Will the last coordinate w' always be 1?

Yes, since we will keep the last row to be "0 0 1"

In general (without homogeneous coordinates)

$$\boldsymbol{x}' = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$$

$\boldsymbol{A}$: 2×2, $\boldsymbol{b}$: 2×1

→ linear transformation + translation.

Matrix Composition

We can combine transformations via matrix multiplication.

A series of transformations:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \equiv \underbrace{\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}}_{T(t_x, t_y)} \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{R(\theta)} \underbrace{\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{S(s_x, s_y)} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

translation rotation scaling

Does order matter?

Yes! Matrix multiplication is not commutative.

Slide credit: A. Efros

What's Preserved With Affine

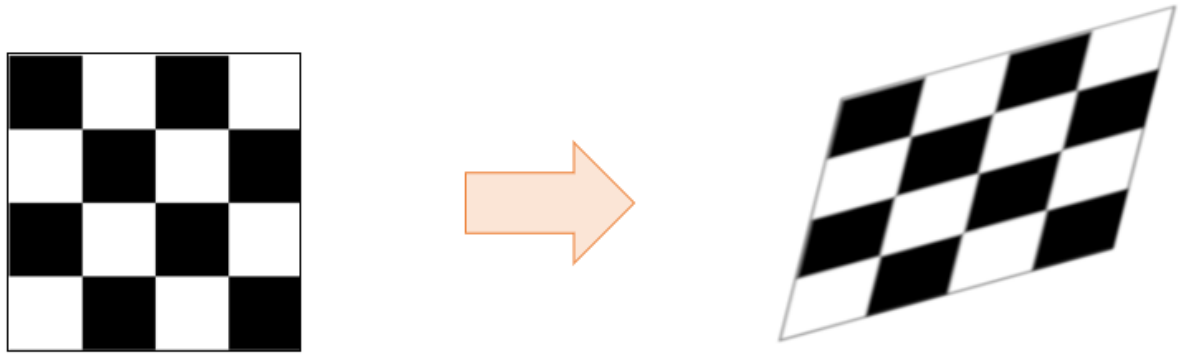
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \equiv \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \equiv \boldsymbol{T} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

After multiplication by T (irrespective of T)

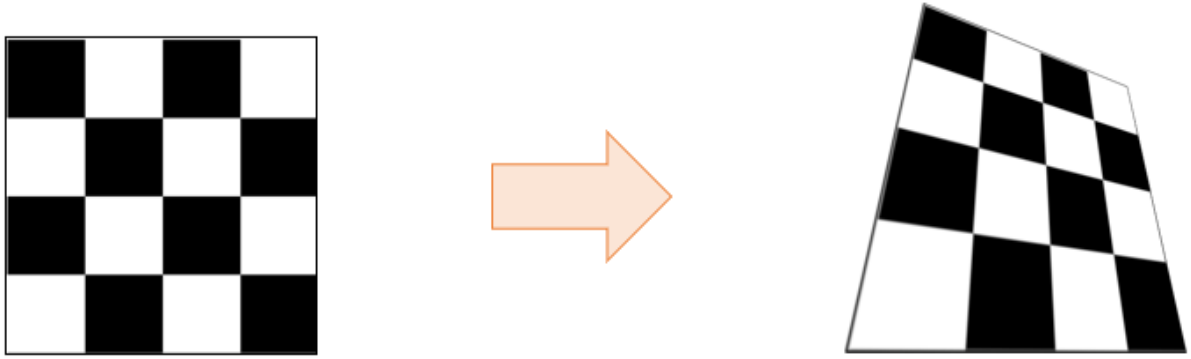
- ~~**Origin is origin: 0 = T0**~~
- Lines are lines
- Parallel lines are parallel

Limitation of Affine Transform

- Affine Transformation is parallel-preserving



- Cannot describe perspective effects where parallel lines are merging to vanishing point!



Perspective Transformations

Set bottom row to not [0,0,1]
Called a perspective/projective transformation or a ***homography***

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \equiv \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

to set this not always 0 0 1

Can compute [x',y',w'] via matrix multiplication.
How do we get a 2D point?

(x'/w', y'/w')

Homogeneous Equivalence

Triple / Equivalent

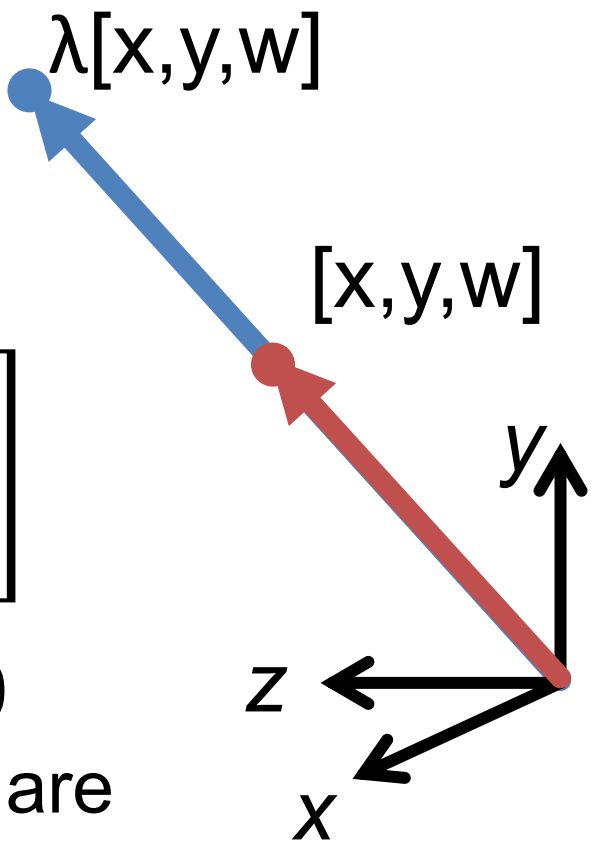
$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} \equiv \begin{bmatrix} u' \\ v' \\ w' \end{bmatrix}$$

$\leftrightarrow$

Double / Equals

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \lambda \begin{bmatrix} u' \\ v' \\ w' \end{bmatrix}$$

$$\lambda \neq 0$$



Two homogeneous coordinates are **equivalent** if they are proportional to each other. **Not = !**

Perspective Transformations

Set bottom row to not [0,0,1]
Called a perspective/projective transformation or a ***homography***

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \equiv \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

How many degrees of freedom?

How Many Degrees of Freedom?

8. since if we now $\|H\| = 1$ and the other 8 element, then we can calculate the rest one.

Can always scale coordinate by non-zero value

Perspective $\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \equiv \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \equiv \frac{1}{i} \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \equiv \frac{1}{i} \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} \equiv \begin{bmatrix} a/i & b/i & c/i \\ d/i & e/i & f/i \\ g/i & h/i & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Homography can always be re-scaled by $\lambda \neq 0$

Typically pick it so last entry is 1 or $\|H\|=1$

What's Preserved With Perspective

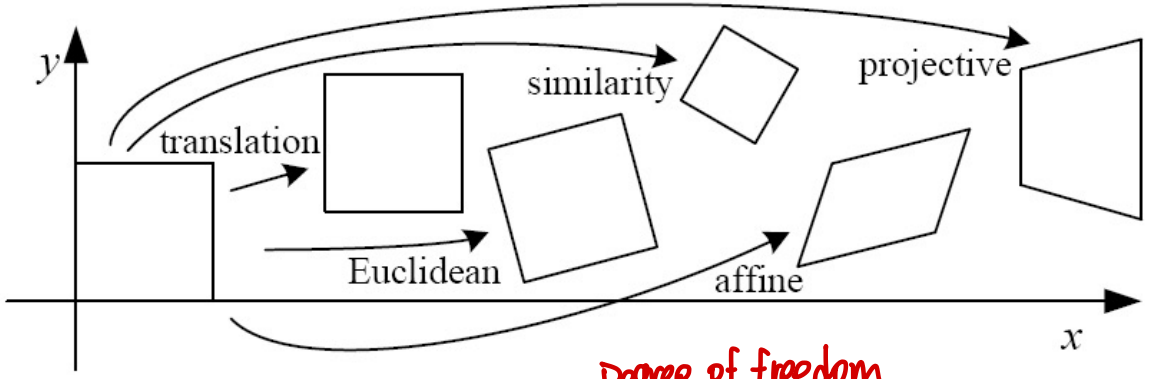
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \equiv \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \equiv \boldsymbol{T} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

After multiplication by T (irrespective of T)

- ~~**Origin is origin: 0 = T0**~~
- Lines are lines
- ~~Parallel lines are parallel~~

Transformation Families

In general: transformations are a nested set of groups



Degree of freedom

Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\left[\begin{array}{c\|c} \boldsymbol{I} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\left[\begin{array}{c\|c} \boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	3	lengths $+\cdots$	
similarity	$\left[\begin{array}{c\|c} s\boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	4	angles $+\cdots$	
affine	$\left[\begin{array}{c} \boldsymbol{A} \end{array}\right]_{2\times 3}$	6	parallelism $+\cdots$	
projective	$\left[\begin{array}{c} \tilde{\boldsymbol{H}} \end{array}\right]_{3\times 3}$	8	straight lines	

rotation + translation

+

=

Diagram credit: R. Szeliski

What Can Homographies Do?

Homography example 1: any two views of a *planar* surface

Figure Credit: S. Lazebnik

What Can Homographies Do?

Homography example 2: any images from two cameras sharing a camera center

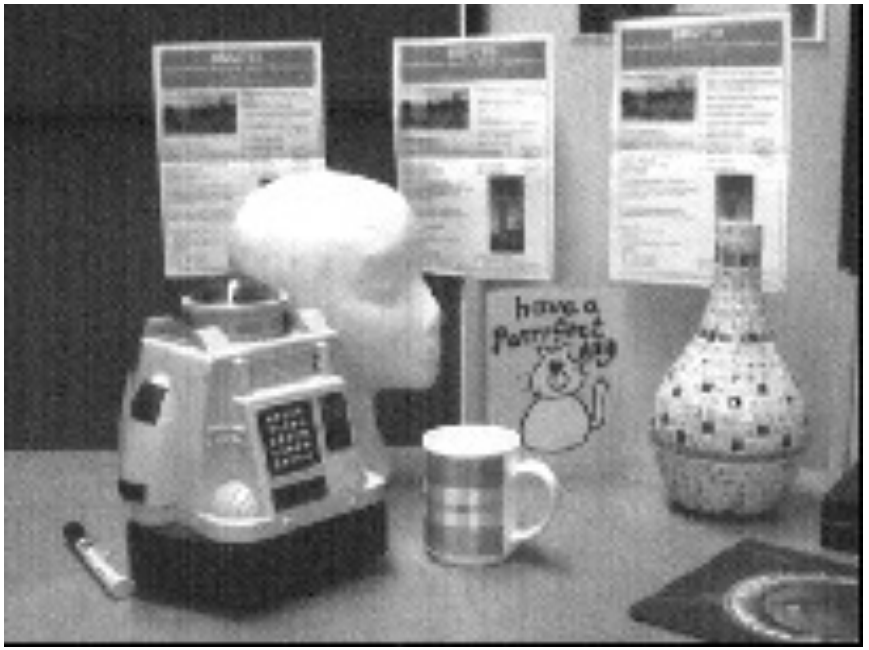
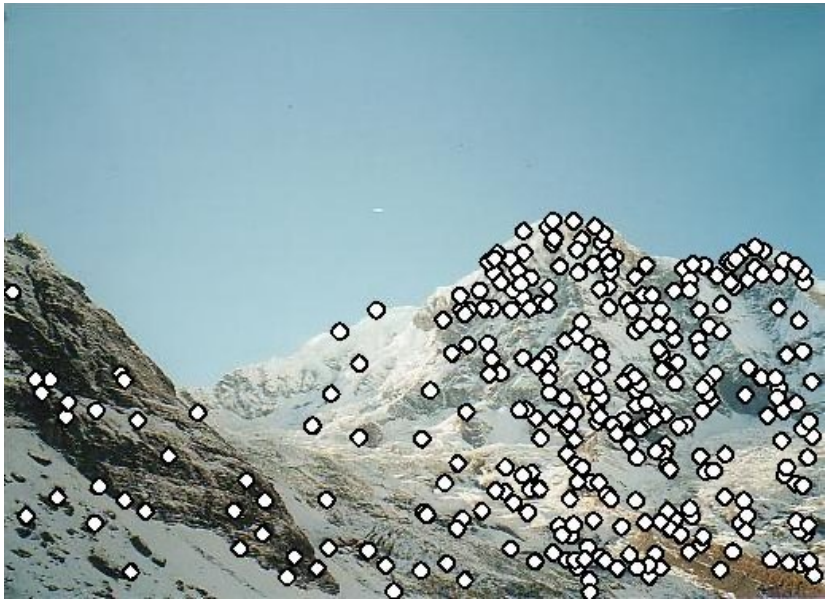


Figure Credit: S. Lazebnik

What Can Homographies Do?

Homography sort of example “3”: far away scene that can be approximated by a plane



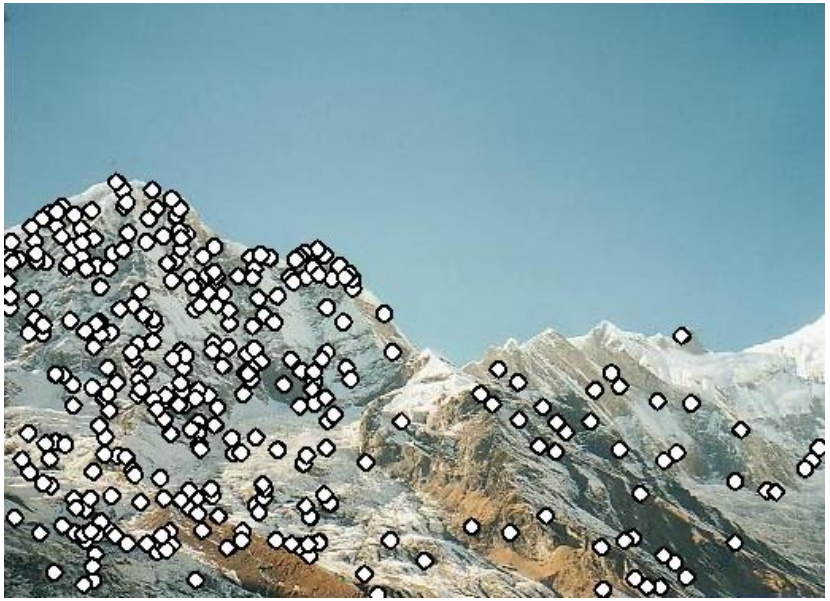


Figure credit: Brown & Lowe

Fun With Homographies

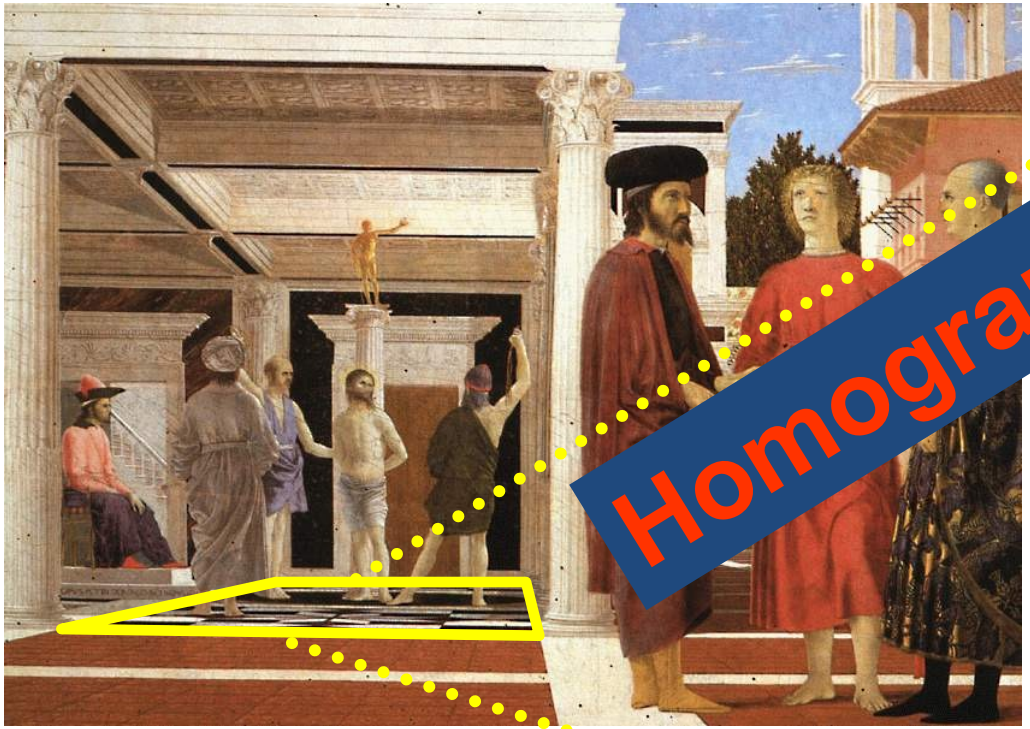
Original image

St. Petersburg
photo by A. Tikhonov

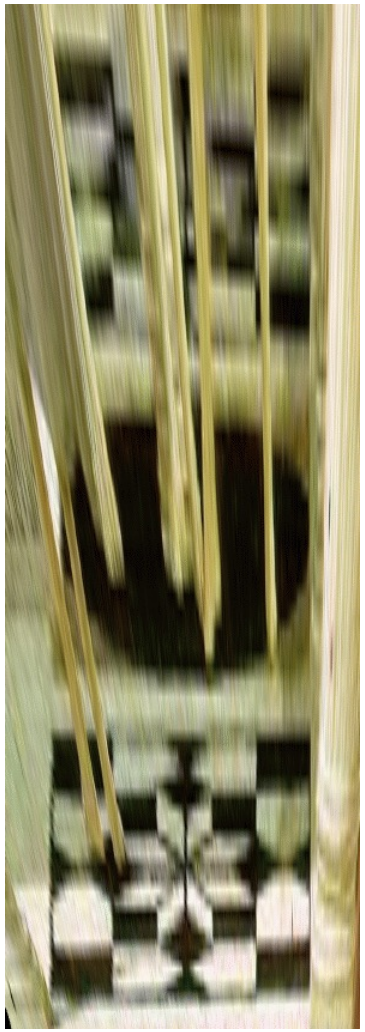
Virtual camera rotations

Slide Credit: A. Efros

Analyzing Patterns



Homography

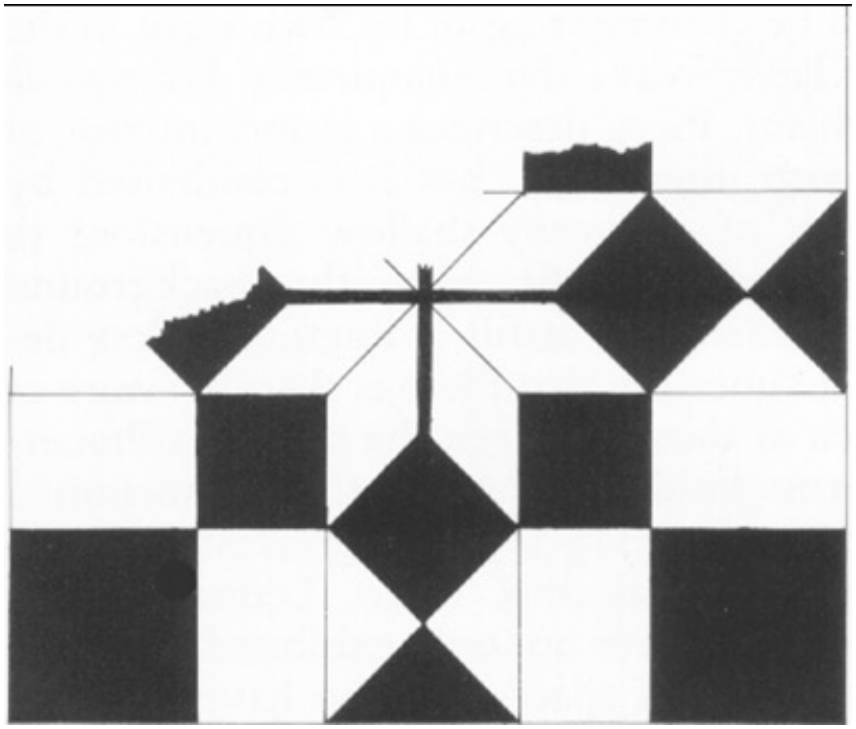


The floor (enlarged)

Automatically rectified floor

Slide from A. Criminisi

Analyzing Patterns



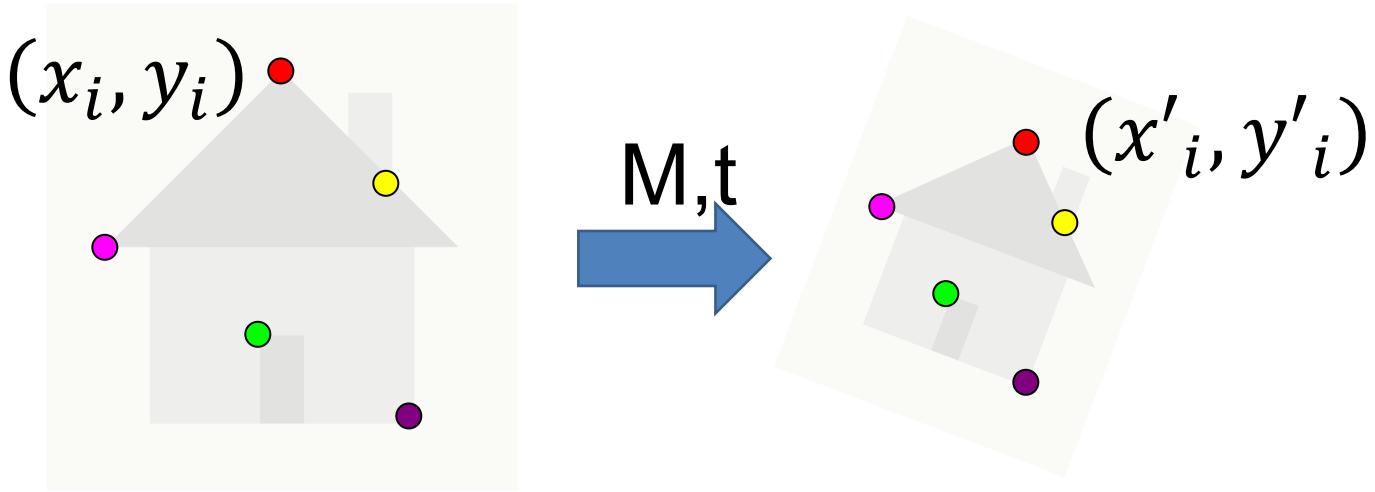
From Martin Kemp *The Science of Art*
(manual reconstruction)

Automatic rectification

Slide from A. Criminisi

Fitting Transformations

Setup: have pairs of correspondences



$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \boldsymbol{M} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \boldsymbol{t}$$

Slide Credit: S. Lazebnik

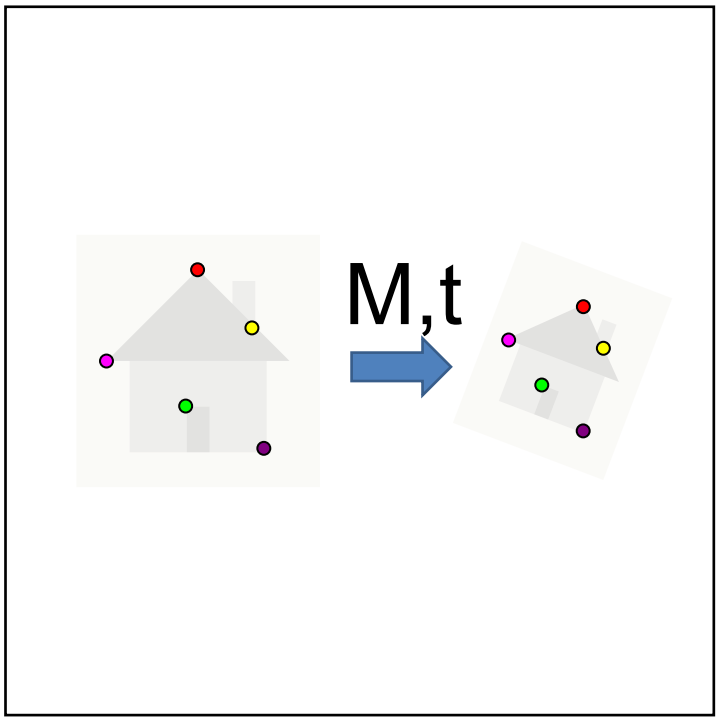
Fitting Transformation

Affine Transformation: M,t

Data: (x_i,y_i,x'_i,y'_i) for $i=1,\ldots,k$

Model:
$[x'_i,y'_i] = \mathbf{M}[x_i,y_i]+\mathbf{t}$

Objective function:
$||[x'_i,y'_i] - (\mathbf{M}[x_i,y_i]+\mathbf{t})||^2$



Fitting Transformations

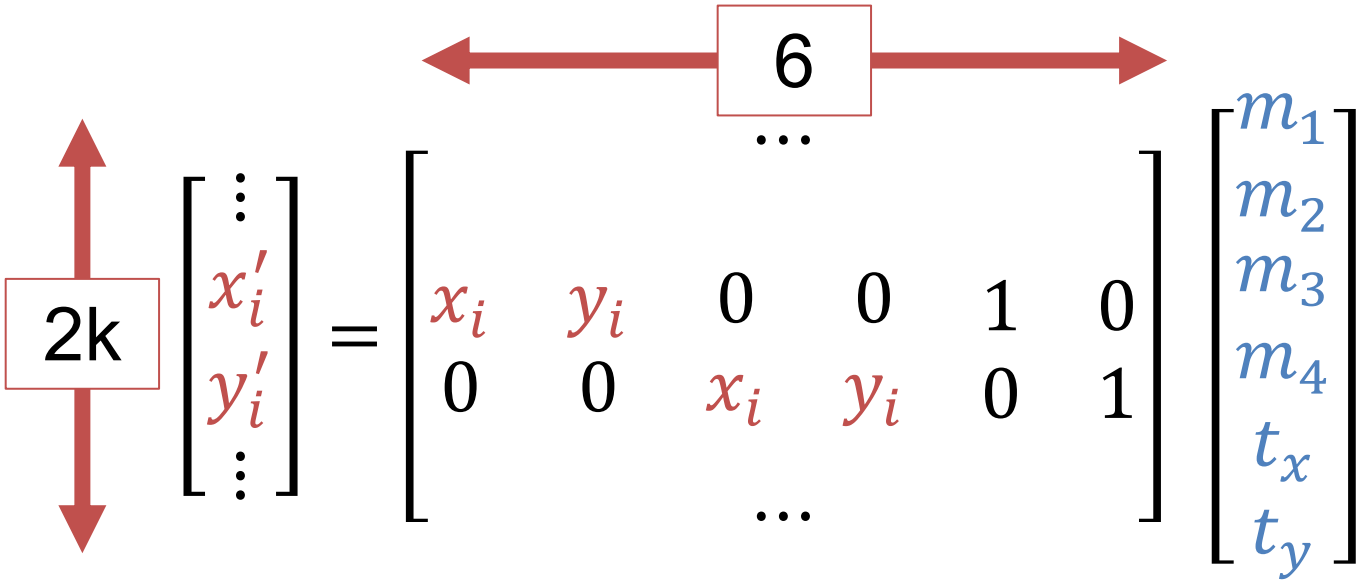
Given correspondences: $[x'_i, y'_i] \leftrightarrow [x_i, y_i]$

$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Set up two equations per point

$$\begin{bmatrix} \vdots \\ x_i' \\ y_i' \\ \vdots \end{bmatrix} = \begin{bmatrix} & & \cdots & & & \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_x \\ t_y \end{bmatrix}$$

Fitting Transformations

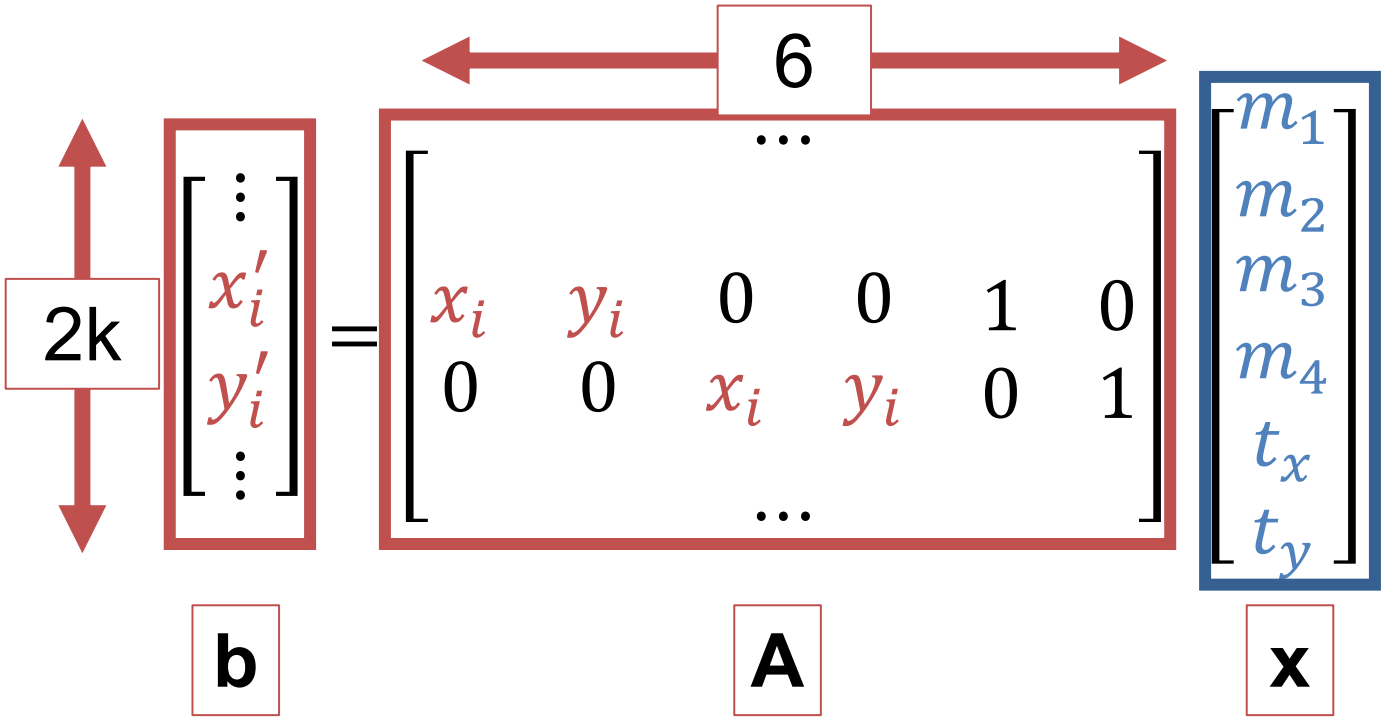


2 equations per point, 6 unknowns

How many points do we need to properly constrain the problem?

#equation = #unknow

Fitting Transformations



Want: **b = Ax** (**x** contains all parameters)

Overconstrained, so solve $\arg\min||\boldsymbol{Ax} - \boldsymbol{b}||$

How?

find x that can min ||Ax-b|| .

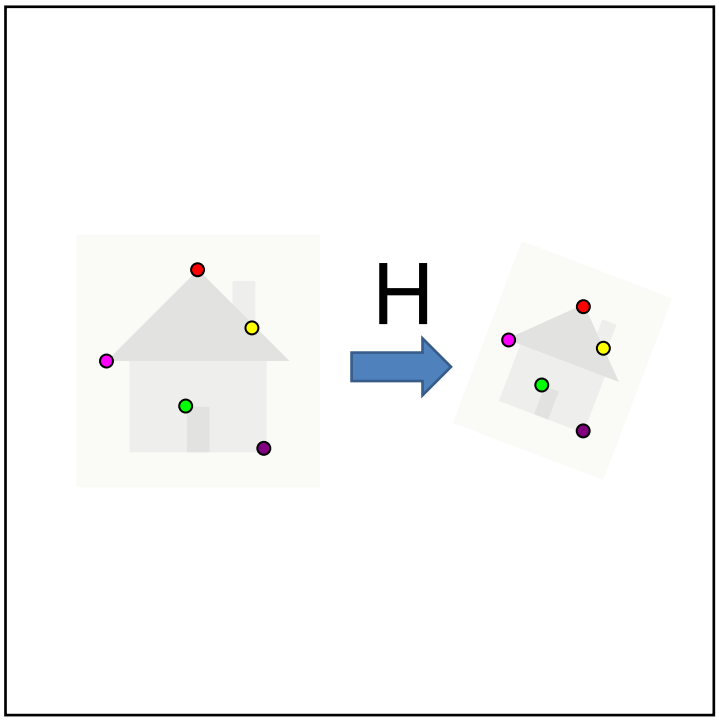
Fitting Transformation

Homography: H

Data: (x_i, y_i, x'_i, y'_i) for $i=1,\ldots,k$

Model:
$[x'_i, y'_i, 1] \equiv \mathbf{H}[x_i, y_i, 1]$

Objective function:
It's complicated



Solving for Homographies

- Transformed points approximate the corresponding points on another image

$$\begin{bmatrix} {x_i}^* \\ {y_i}^* \\ w \end{bmatrix} \cong \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \equiv \boldsymbol{T} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

Solving for Homographies

- Transformed points approximate the corresponding points on another image

I define $x_i' = x_i^*/w$.

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} = \begin{bmatrix} x_i^*/w \\ y_i^*/w \\ 1 \end{bmatrix} \cong \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$x_i' \approx \frac{ax_i + by_i + c}{gx_i + hy_i + i}$$

$$y_i' \approx \frac{dx_i + ey_i + f}{gx_i + hy_i + i}$$

Solving for Homographies

- Transformed points approximate the corresponding points on another image

$$\begin{bmatrix} x_i'/w \\ y_i'/w \\ 1 \end{bmatrix} \cong \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$x_i'(gx_i + hy_i + i) \approx ax_i + by_i + c$$

$$y_i'(gx_i + hy_i + i) \approx dx_i + ey_i + f$$

Solving for Homographies

$$x_i'(gx_i + hy_i + i) = ax_i + by_i + c$$

$$y_i'(gx_i + hy_i + i) = dx_i + ey_i + f$$

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x_i'x_i & -x_i'y_i & -x_i' \\ 0 & 0 & 0 & x_i & y_i & 1 & -y_i'x_i & -y_i'y_i & -y_i' \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Solving for Homographies

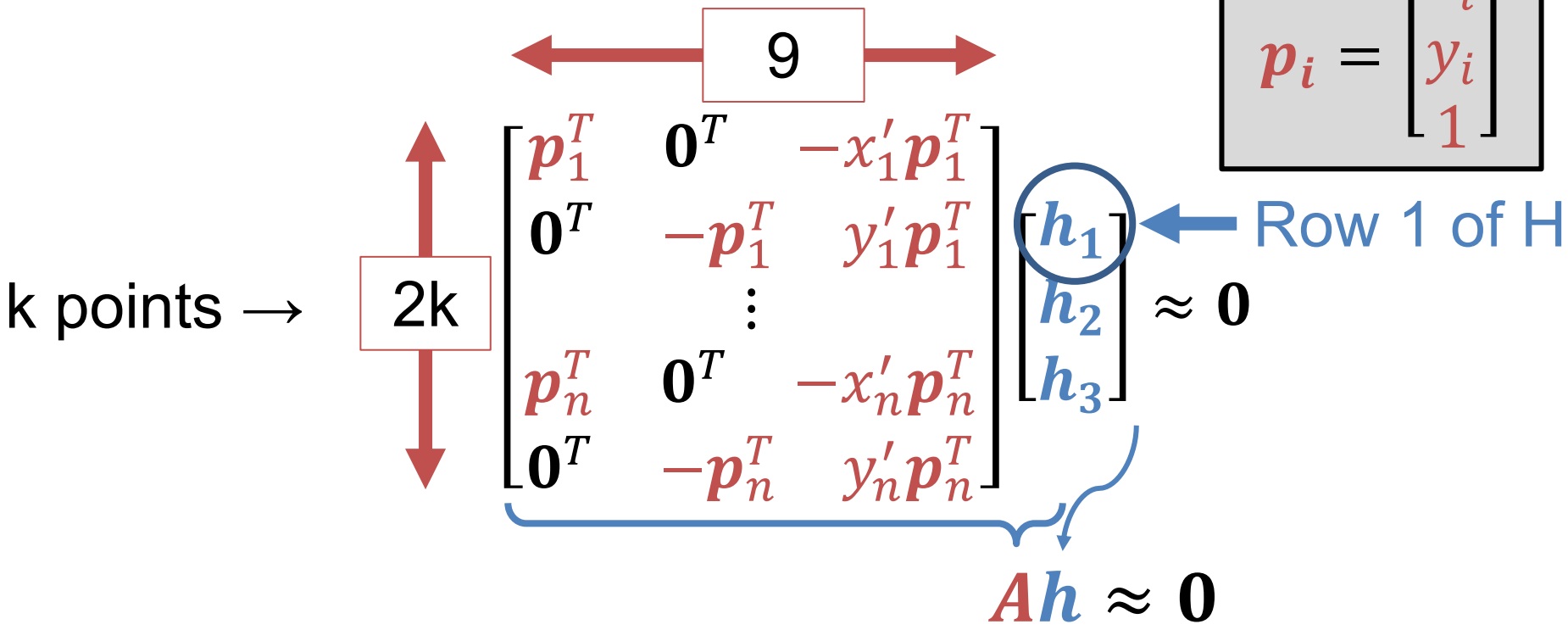
$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1'x_1 & -y_1'y_1 & -y_1' \\ & & & & & \vdots & & & \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$$

A: 2k x 9

$\boldsymbol{h}$: 9

Fitting Transformation

$$\boldsymbol{p}_i = \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

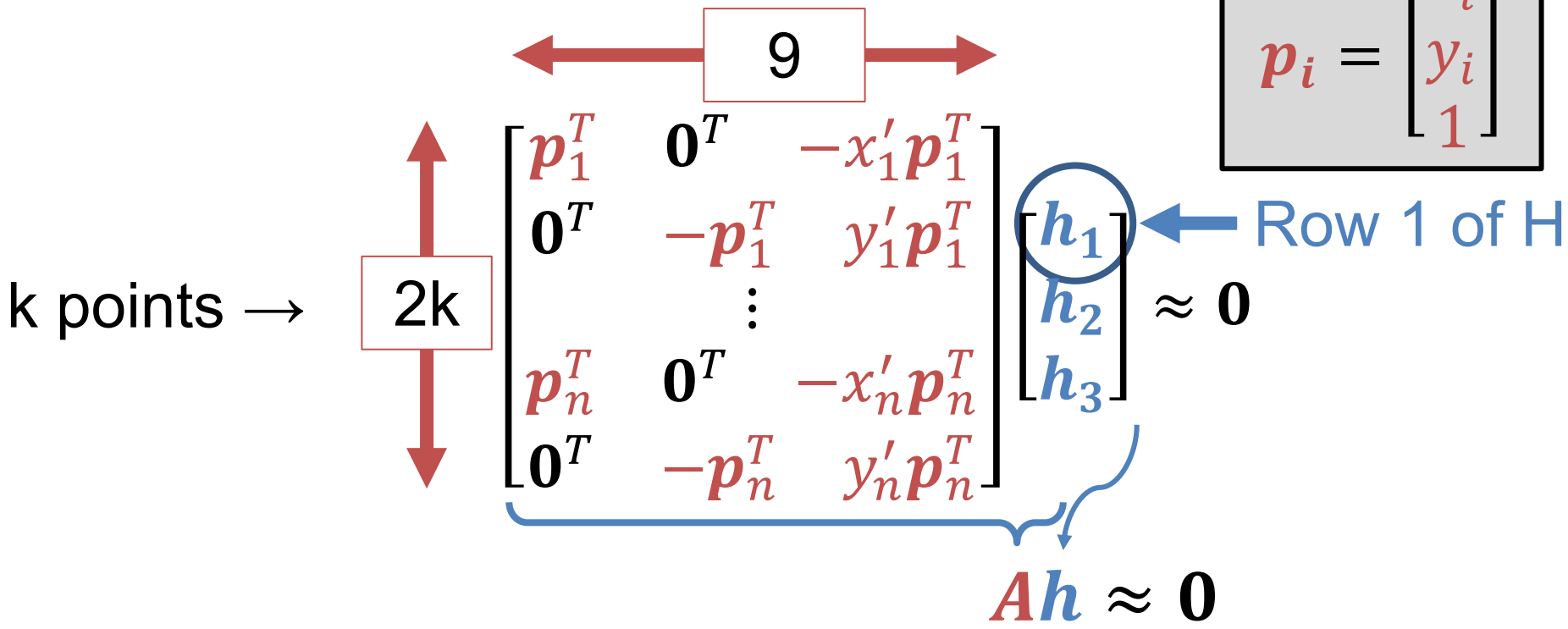


Homogeneous coordinates => h can have any scale

Therefore: Solve for Unit vector h

Fitting Transformation

$$\boldsymbol{p}_i = \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

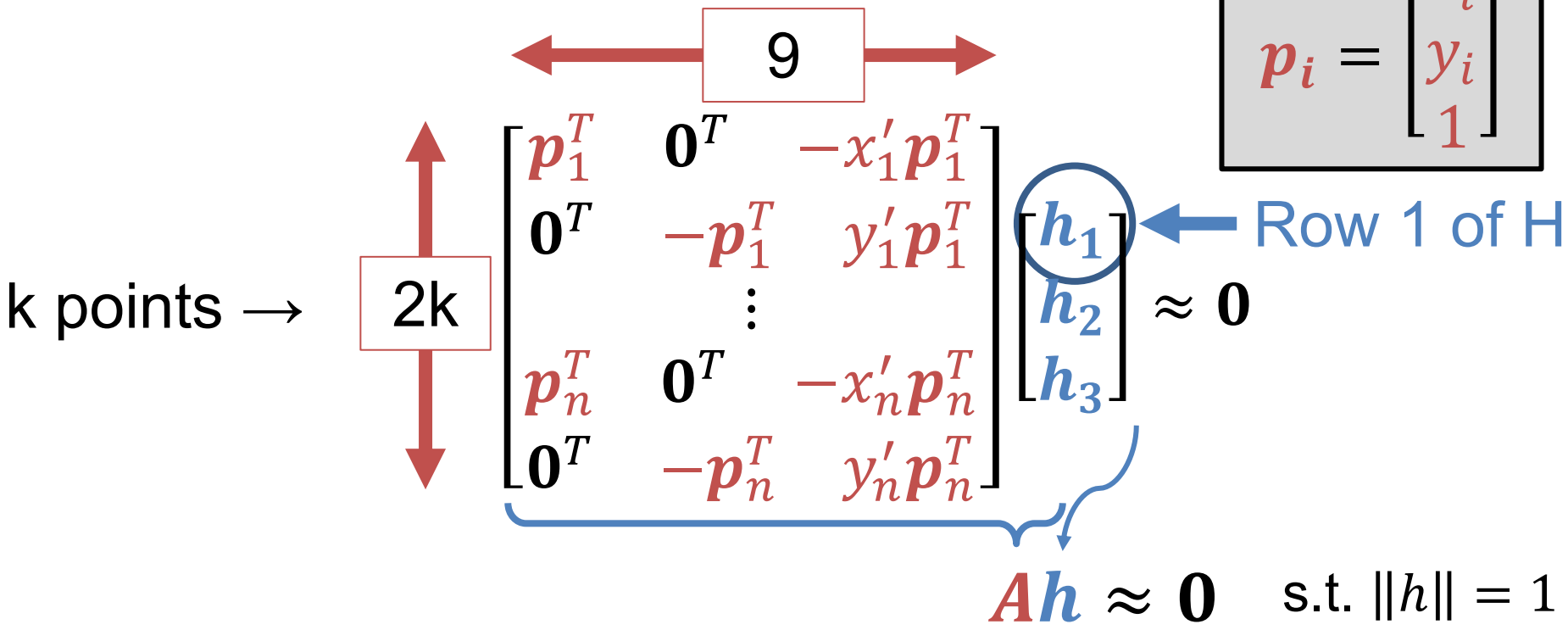


Homogeneous coordinates => h can have any scale

Therefore: Solve for Unit vector h

Fitting Transformation

$$p_i = \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$



What do we use from last time?

$$h^* = \arg\min_{\|h\|=1} \|Ah\|^2$$

constrain

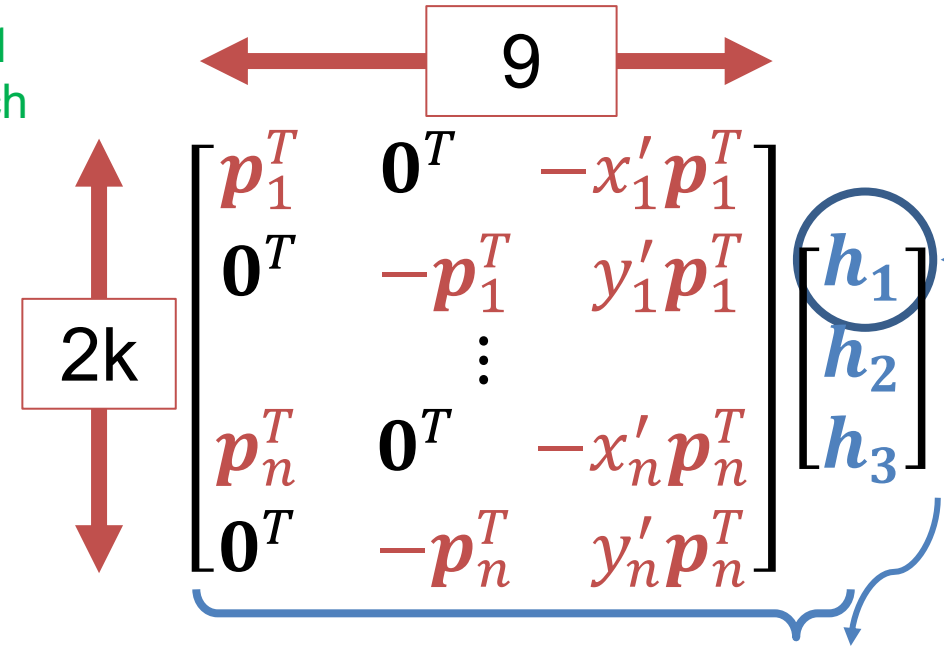
→ Eigenvector of A^TA with smallest eigenvalue

Fitting Transformation

$$p_i = \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

Note: multiplying -1 to all entries in each row of A will not change the output of A^TA. Why?

k points →



Row 1 of H

$$\boldsymbol{Ah} \approx \mathbf{0} \quad \text{s.t. } \|h\| = 1$$

What do we use from last time?

$$h^* = \arg\min_{\|h\|=1} \|Ah\|^2$$

→ Eigenvector of A^TA with smallest eigenvalue

Smallest Eigenvector of A^TA

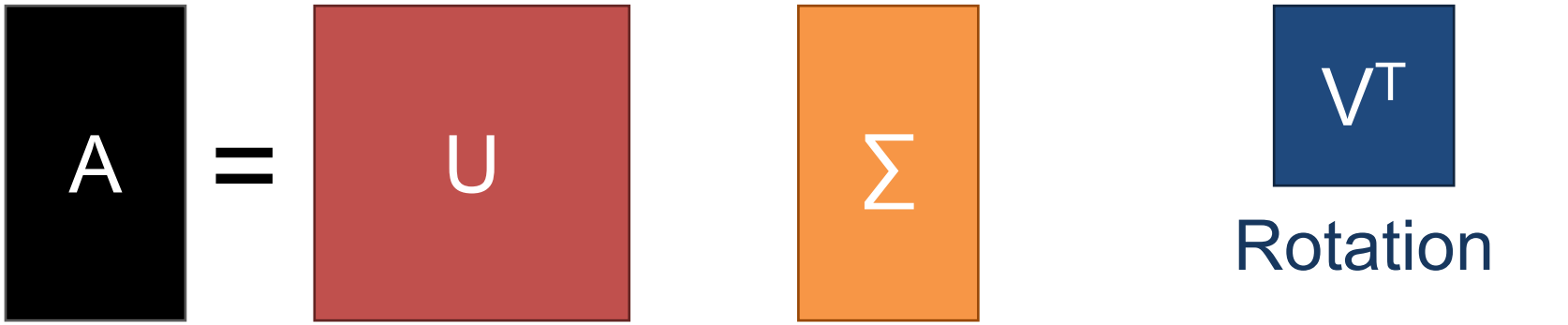
2 Ways of getting this:

Calculate all eigenvalue.

1. Eigenvalue decomposition of A^TA
2. Singular Value Decomposition (SVD)

The Singular Value Decomposition

Can **always** write a mxn matrix **A** as: $\boldsymbol{A = U\Sigma V^T}$



Smallest Eigenvector of A^TA

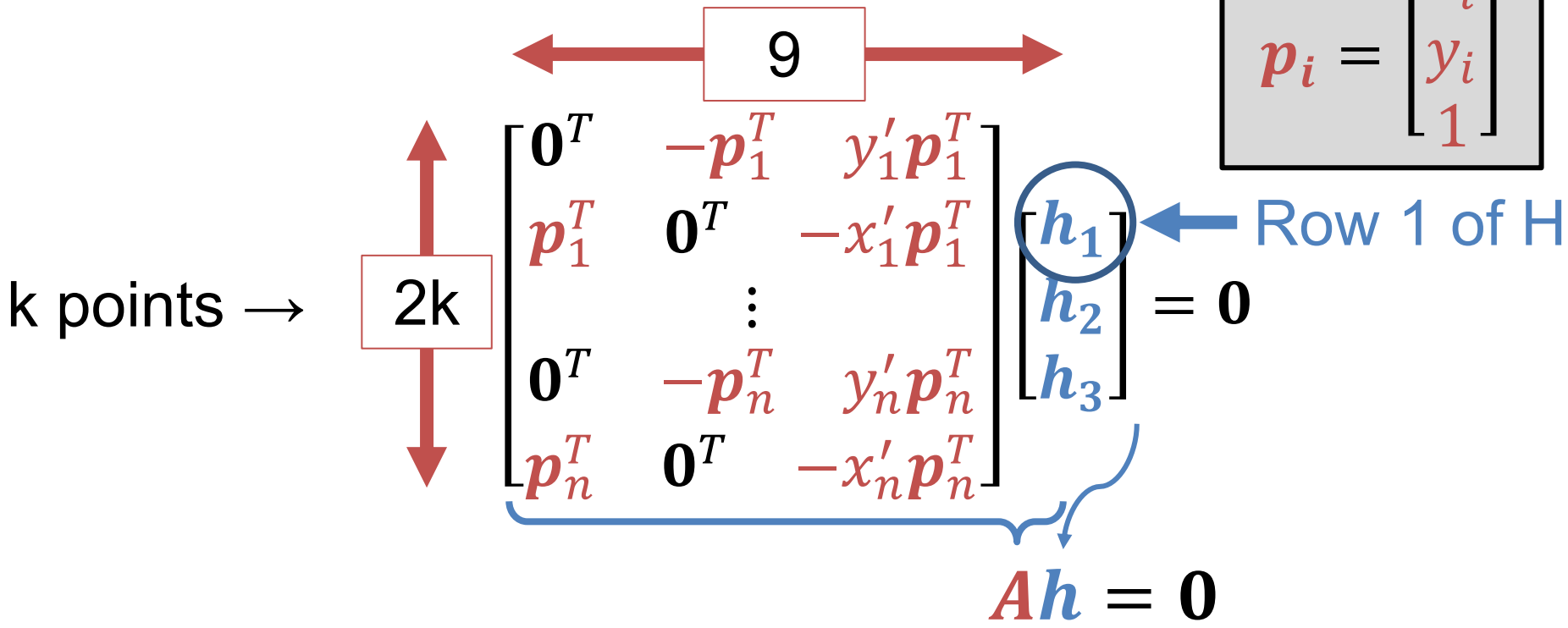
2 Ways of getting this: both are fine!

1. Eigenvalue decomposition of A^TA
   -- Could be simpler (np.linalg.eig)
2. Singular Value Decomposition (SVD)

-- SVD is can be more efficient and numerically stable

-- But both are viable options!

In Practice



Should consist of lots of {x,y,x',y',0, and 1}.

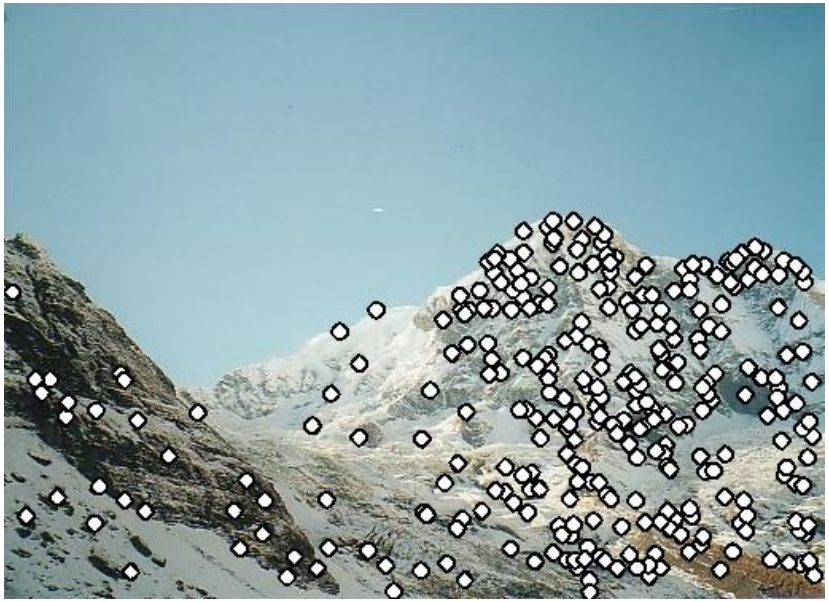
If it fails, **assume** you mistyped.

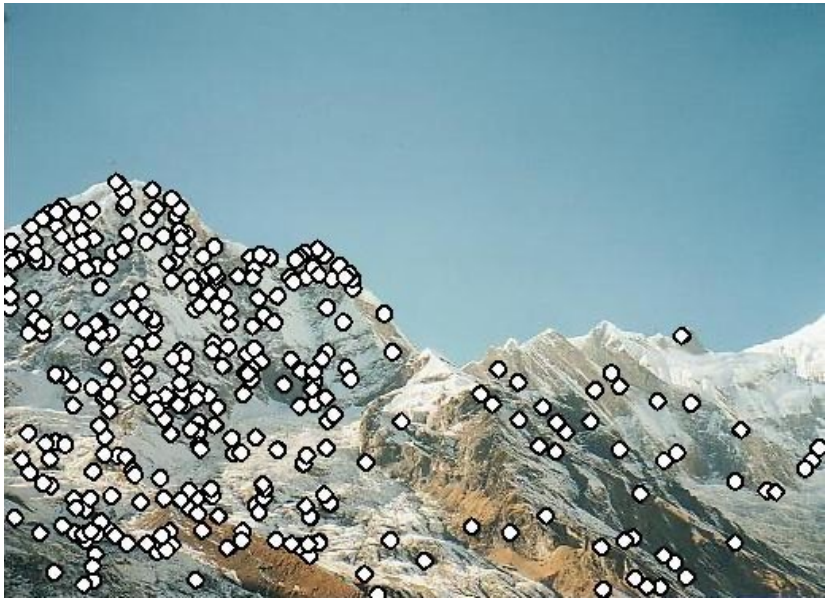
Re-type differently and compare all entries.

Debug first to make sure all entries are correct.

RANSAC

Many (most) detected correspondences are false: RANSAC





RANSAC

Many (most) detected correspondences are false: RANSAC

Remember that we select a subset with the **smallest possible** number of data

What should that be for Homographies?

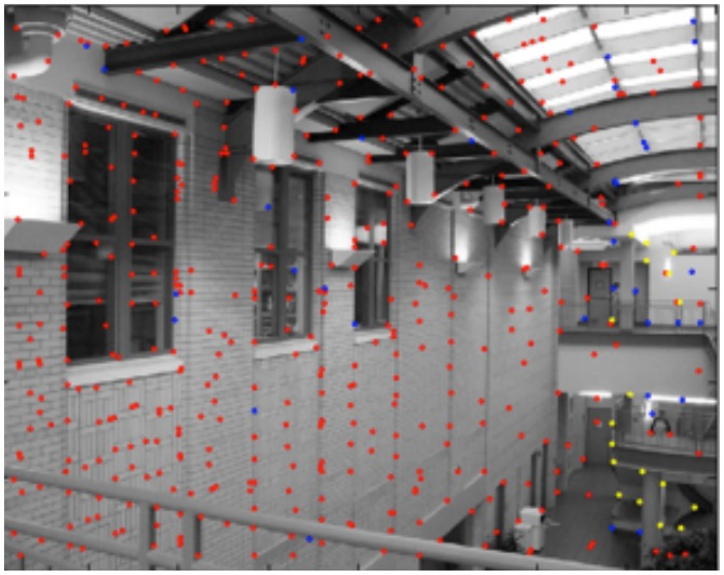
Answer: 4 matched pairs. Why?

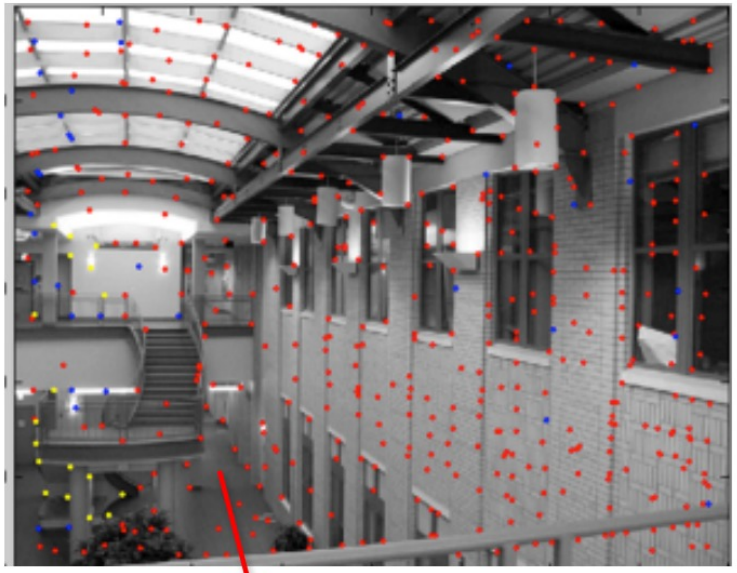
$$x_i'(gx_i + hy_i + i) = ax_i + by_i + c$$

$$y_i'(gx_i + hy_i + i) = dx_i + ey_i + f$$

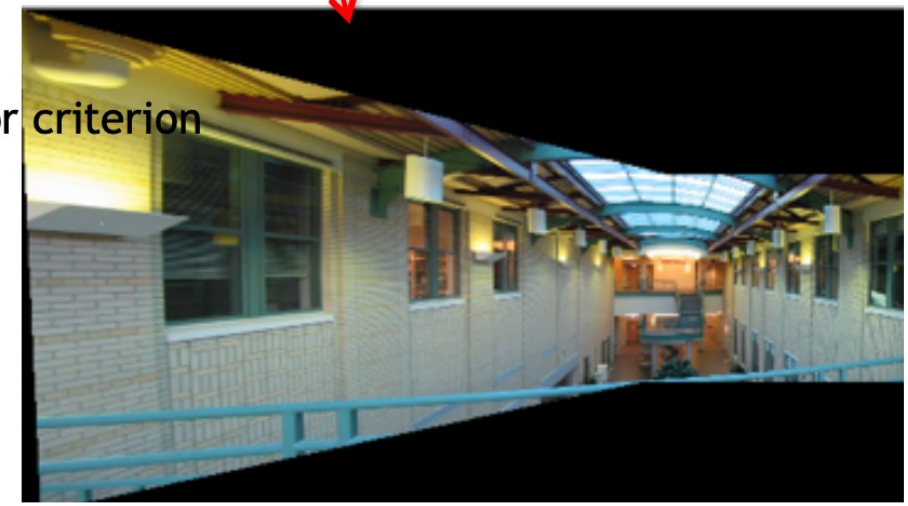
2 equations per point.

RANSAC



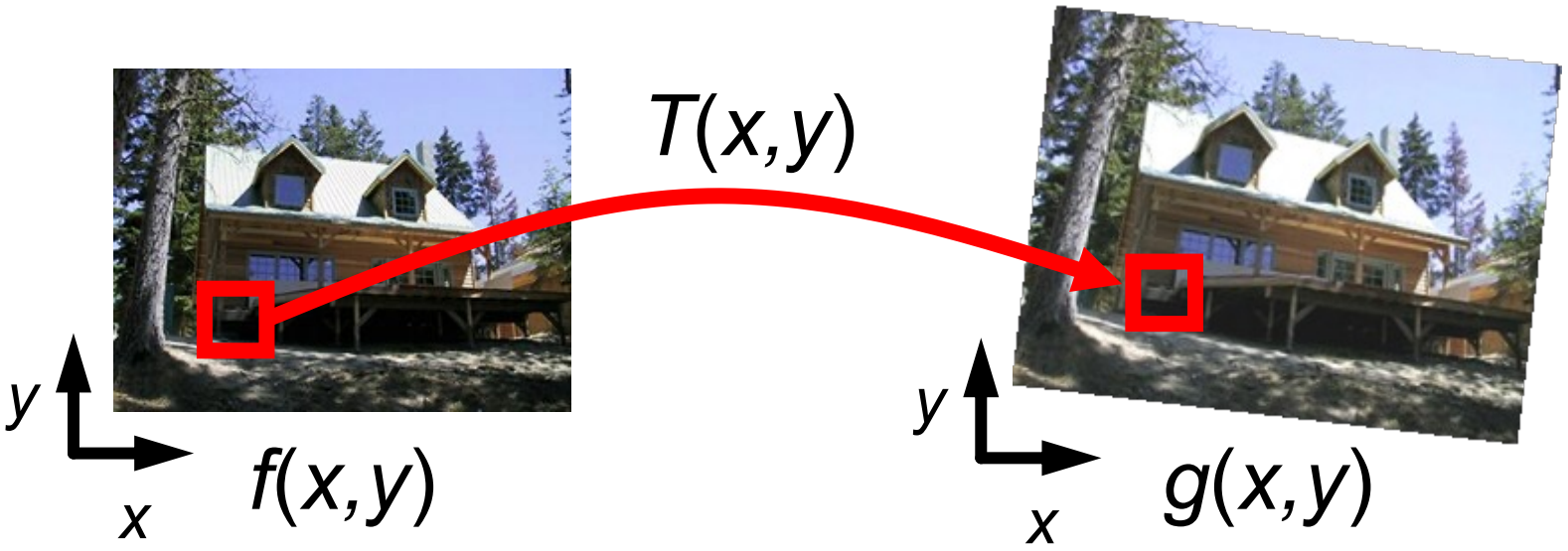


Red:
rejected by 2nd nearest neighbor criterion
Blue:
Ransac outliers
Yellow:
inliers



Slide Credit: S. Seitz

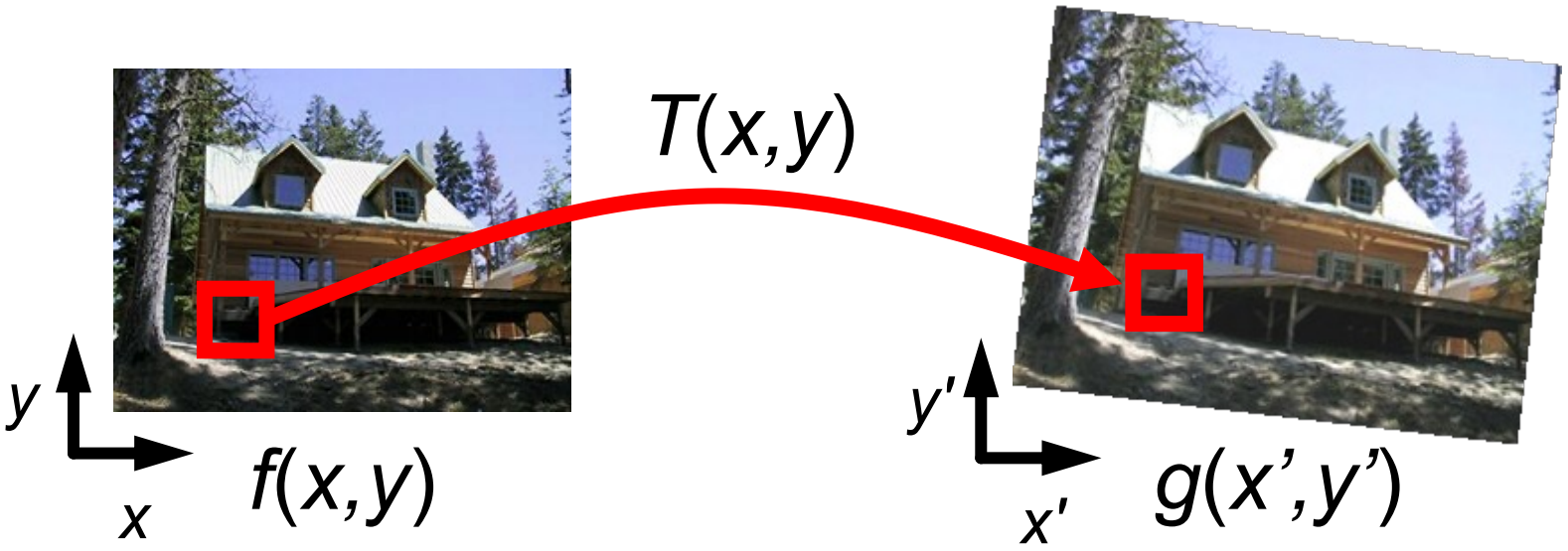
Image Warping



Given a coordinate transform *(x',y')* = *T*(*x*,*y*) and a source image *f*(*x*,*y*), how do we compute a transformed image *g*(*x'*,*y'*) = *f*(*T*(*x*,*y*))?

Slide Credit: A. Efros

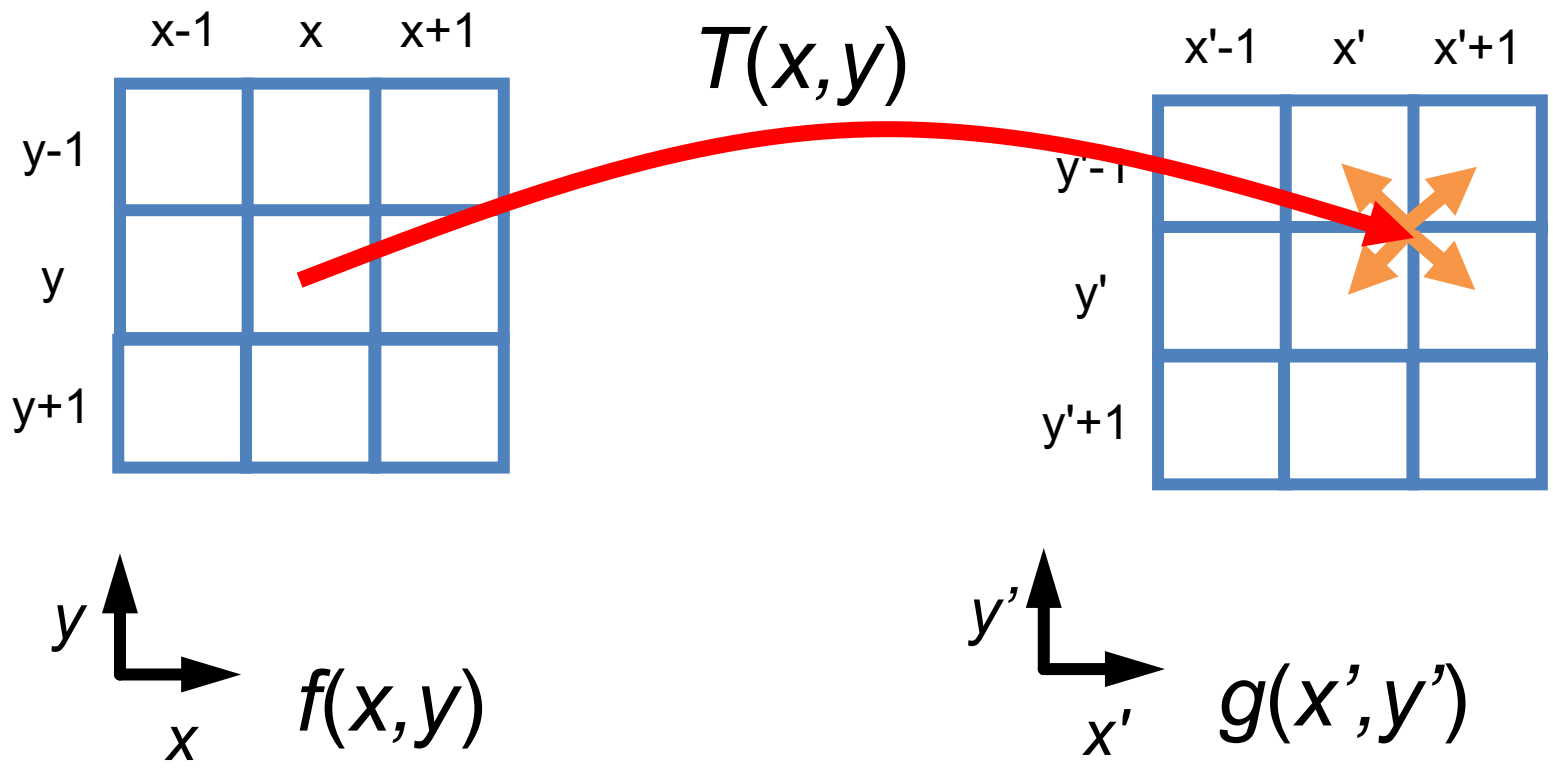
Forward Warping



Send the value at each pixel (x,y) to
the new pixel (x',y') = T([x,y])

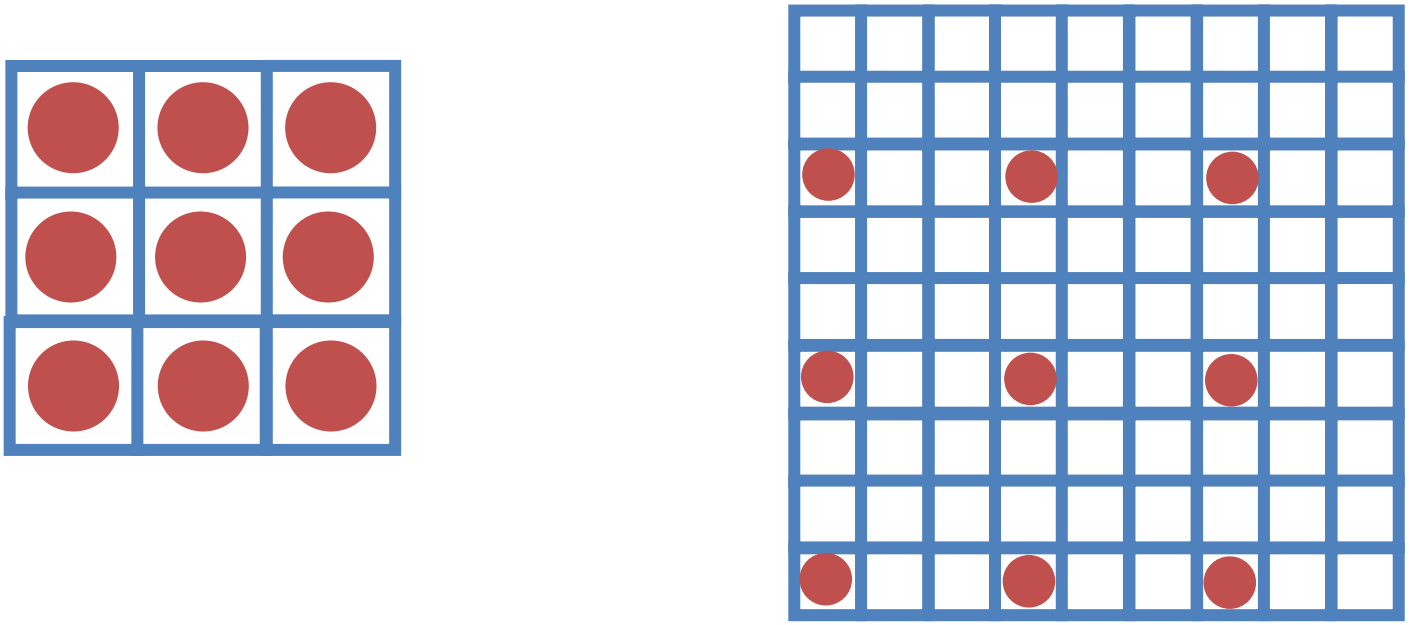
Slide Credit: A. Efros

Forward Warping



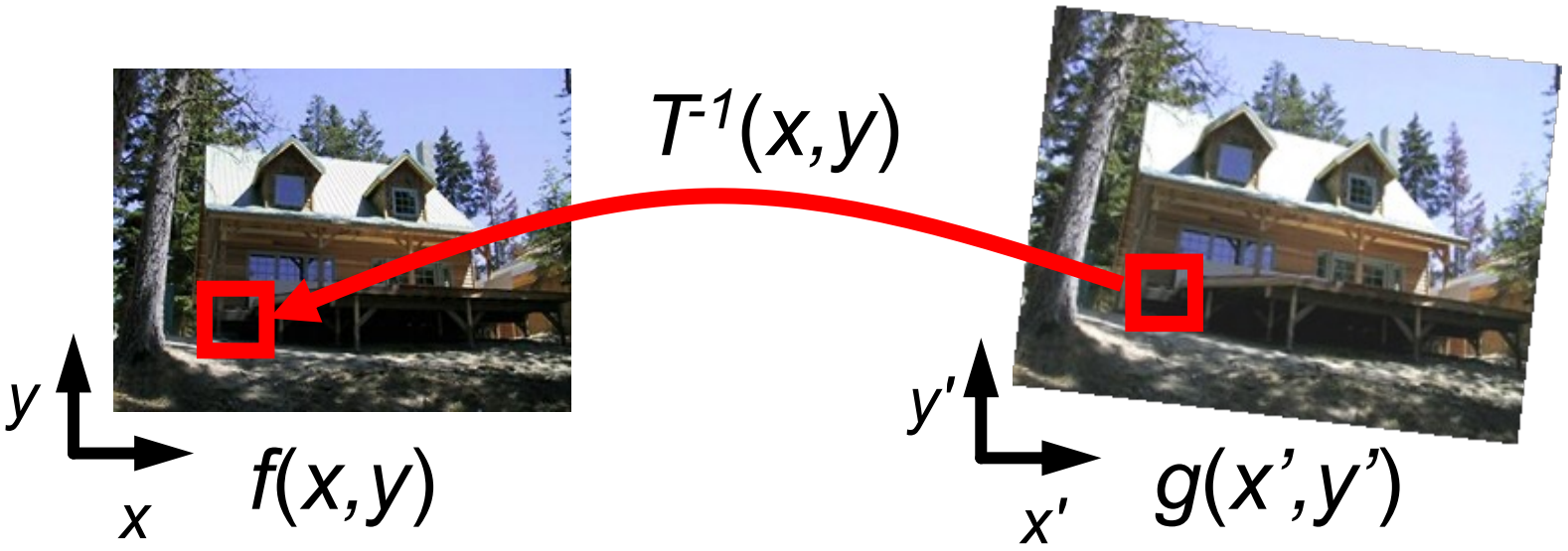
If you don't hit an exact pixel, give the value to each of the neighboring pixels ("splatting").

Forward Warping



Suppose T(x,y) scales by a factor of 3.
Hmmmm.

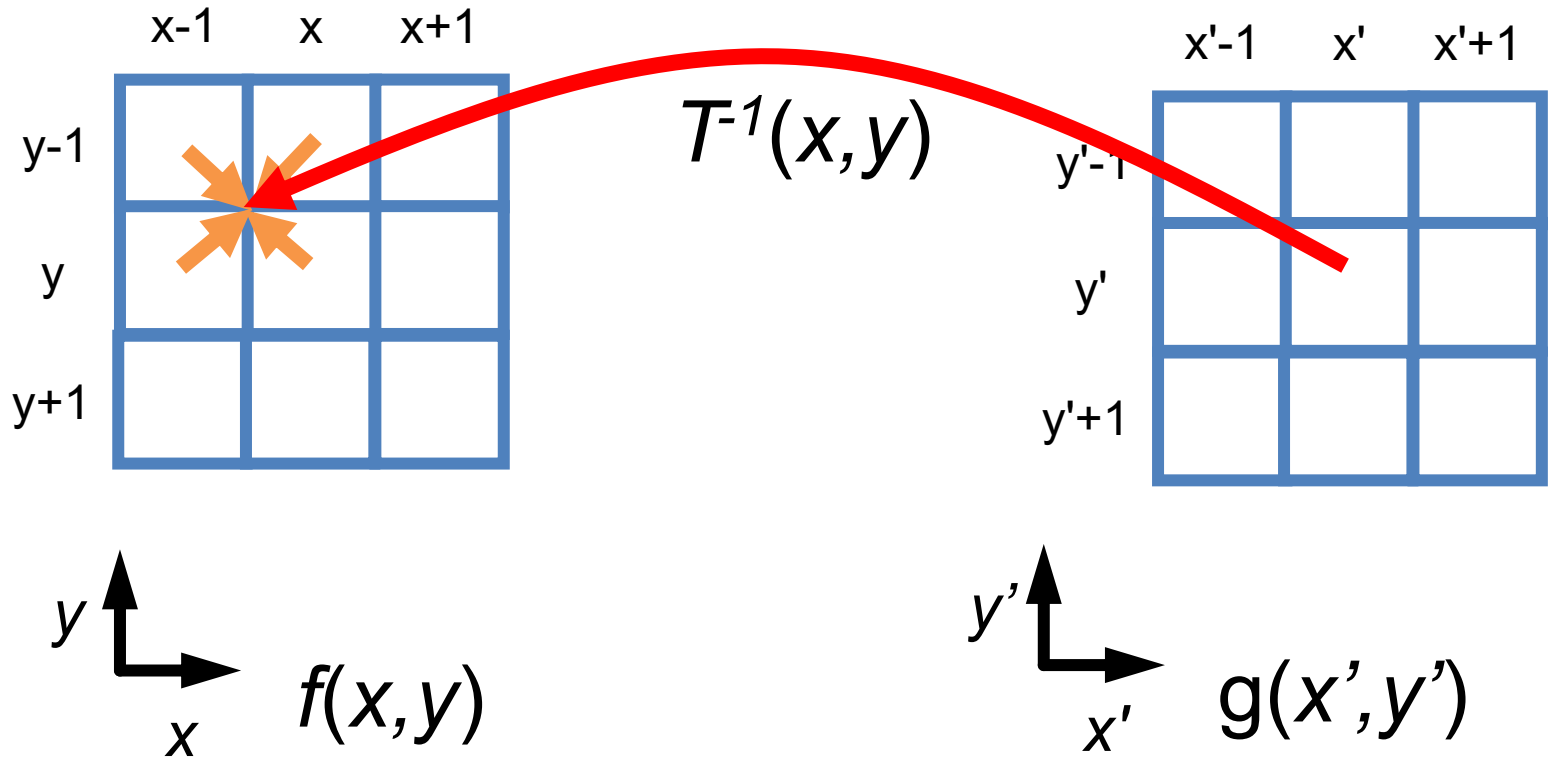
Inverse Warping



Find out where each pixel g(x',y') should get its value from, and steal it.
Note: requires ability to invert T

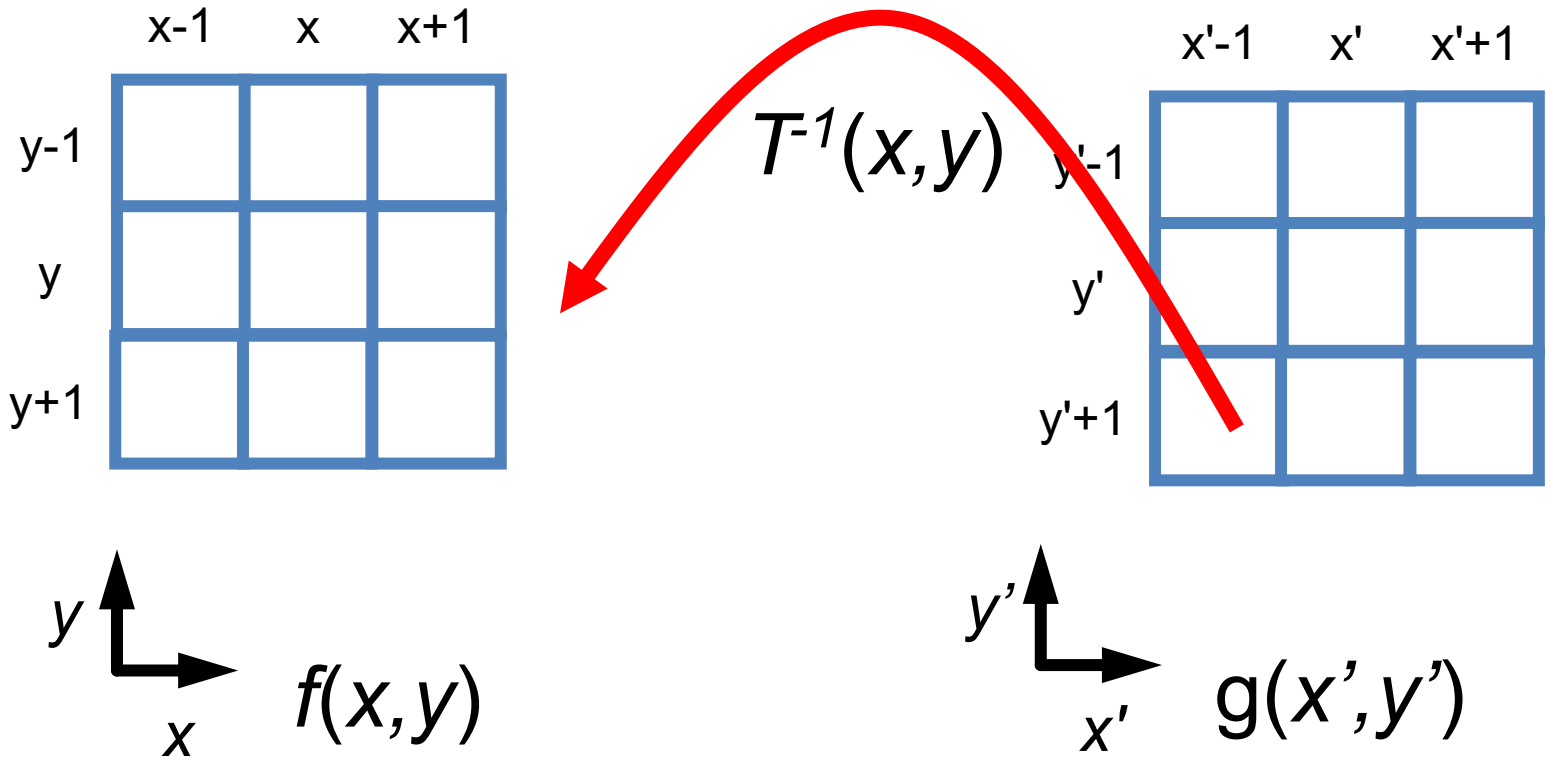
Slide Credit: A. Efros

Inverse Warping



If you don't hit an exact pixel, figure out how to take it from the neighbors.

Inverse Warping



If it goes out of bounds, set it to be zero

Mosaicing

Warped Input 1 I_1

Warped Input 2 I_2

Can warp an image. Pixels that don't have a corresponding pixel in the image are set to a chosen value (often 0)

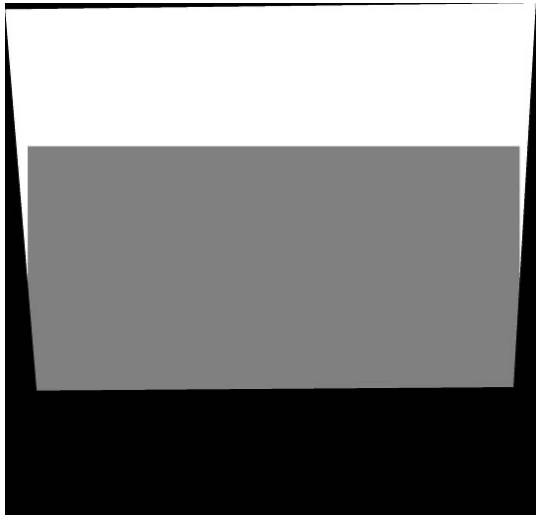
Image Credit: A. Efros

Mosaicing

Warped Input 1 I_1

Warped Input 2 I_2

α



$\alpha I_1 + (1-\alpha) I_2$

Image Credit: A. Efros

Putting it Together

How do you make a panorama?

Step 1: Find “features” to match

Step 2: Describe Features

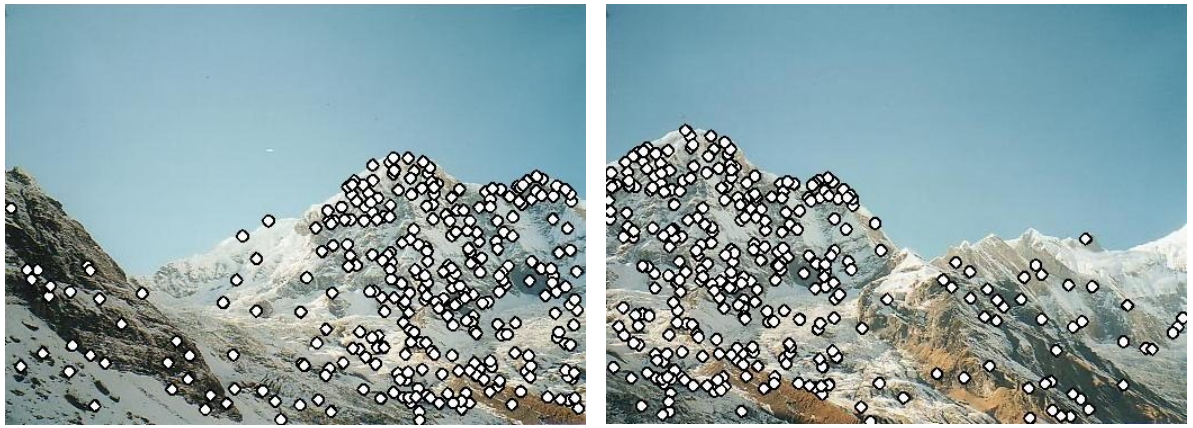
Step 3: Match by Nearest Neighbor

Step 4: Fit H via RANSAC

Step 5: Blend Images

Putting It Together 1

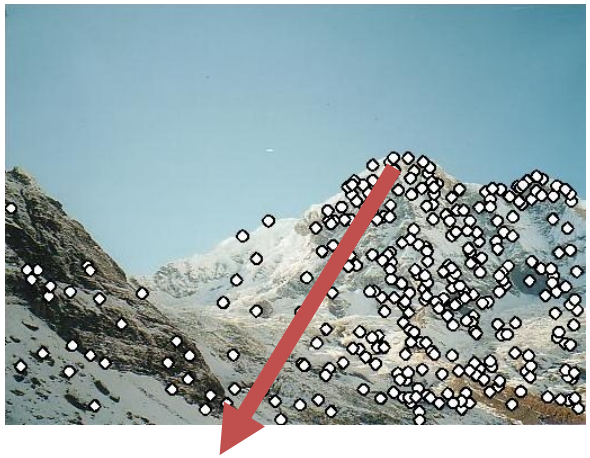
Find corners/blobs

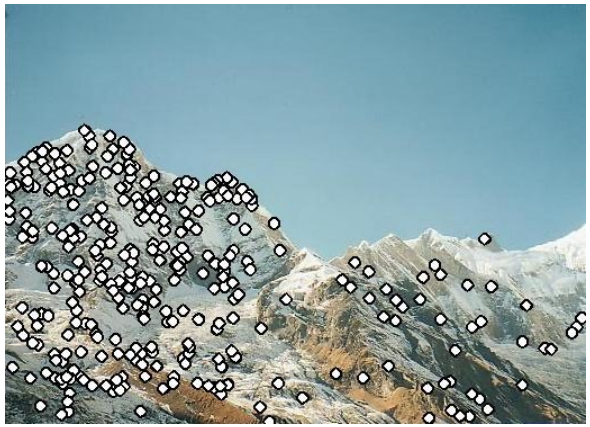


- (Multi-scale) Harris; or
- Laplacian of Gaussian

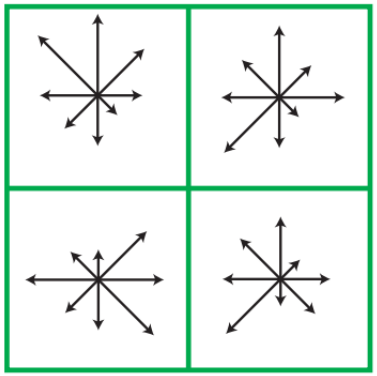
Putting It Together 2

Describe Regions Near Features





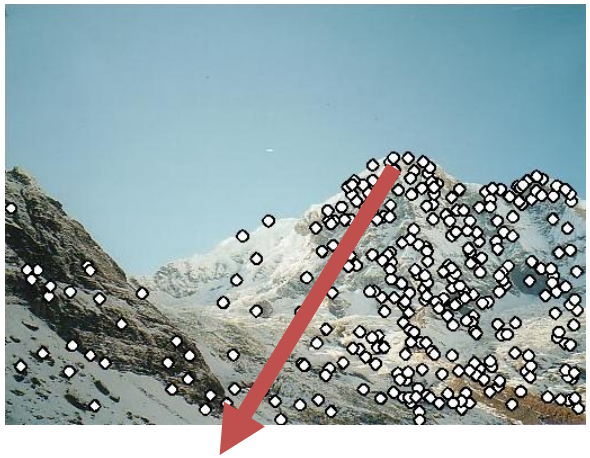
$x_q \in R^{128}$

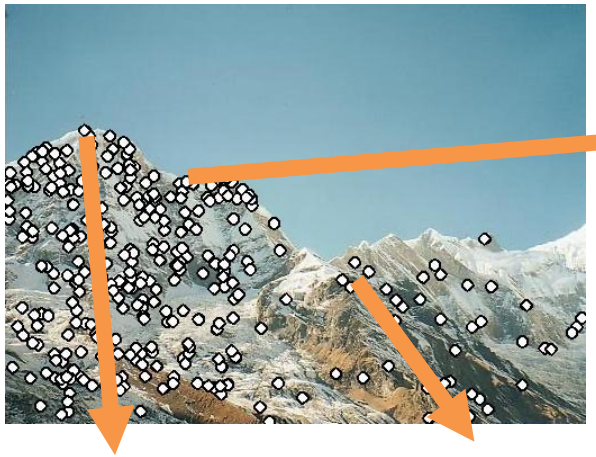


Build histogram of gradient orientations (SIFT) *(But in practice use opencv)*

Putting It Together 3

Match Features Based On Region





$x_3 \in R^{128}$

$x_q \in R^{128}$ $x_1 \in R^{128}$ $x_2 \in R^{128}$

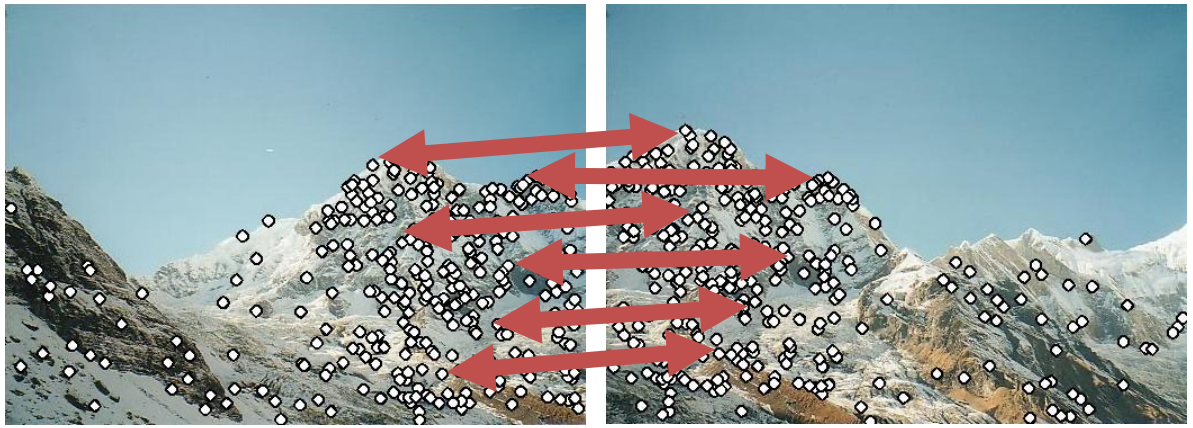
Sort by distance to: x_q $\|x_q - x_1\| < \|x_q - x_2\| < \|x_q - x_3\|$

Accept match if: $\|x_q - x_1\| / \|x_q - x_2\|$

Nearest neighbor is far closer than 2nd nearest neighbor

Putting It Together 4

Fit transformation H via RANSAC



for trial in range(Ntrials):
    Pick sample
    Fit model
    Check if more inliers
Re-fit model with most inliers

$$\arg\min_{\|\boldsymbol{h}\|=1} \|\boldsymbol{A}\boldsymbol{h}\|^2$$

Putting It Together 5

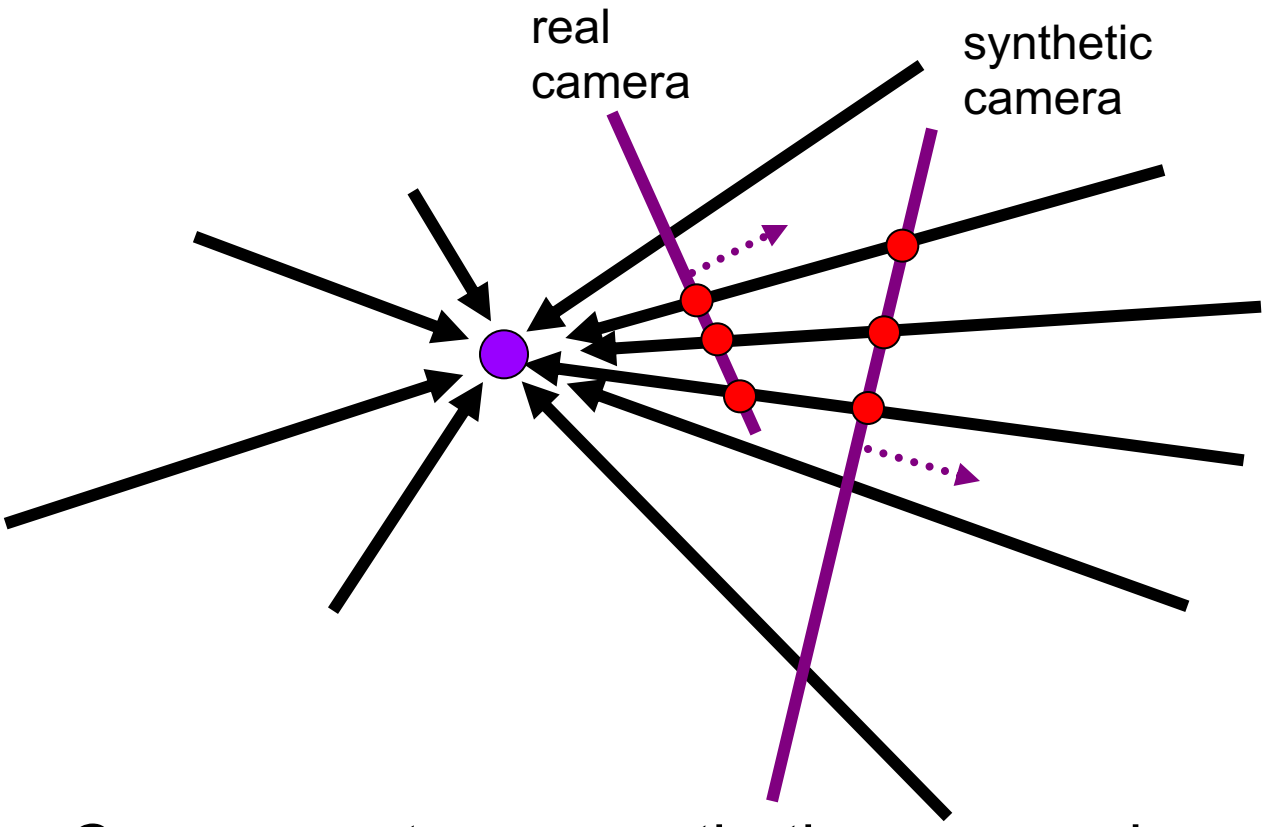
Warp images together

Resample images with inverse warping and blend

(but in practice, just call opencv for inverse warping)

For the curious

A pencil of rays contains all views



Can generate any synthetic camera view
as long as it has **the same center of projection**!

Slide Credit: A. Efros

Bonus Art

Analyzing Patterns

***St. Lucy Altarpiece,* D. Veneziano**

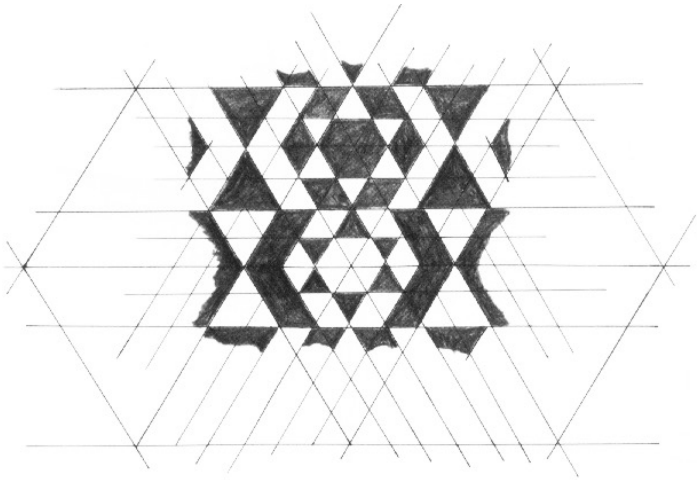
What is the (complicated) shape of the floor pattern?

Automatically rectified floor

Slide from A. Criminisi

Analyzing Patterns

Automatic rectification



From Martin Kemp, *The Science of Art*
(manual reconstruction)

Slide from A. Criminisi

Simplification: Two-band Blending

- Brown & Lowe, 2003
  - Only use two bands: high freq. and low freq.
  - Blend low freq. smoothly
  - Blend high freq. with no smoothing: binary alpha

Figure Credit: Brown & Lowe

2-band “Laplacian Stack” Blending

Low frequency (λ > 2 pixels)

High frequency (λ < 2 pixels)

Linear Blending

2-band Blending